NÉLSON CASARI

dBASE II PLUS PARA MSX PROGRAMÁVEL

atlas

dBASE II PLUS PARA MSX

PROGRAMÁVEL

NÉLSON CASARI

dBASE II PLUS PARA MSX PROGRAMÁVEL

CURSO PRÁTICO

2ª Edição

SÃO PAULO
EDITORA ATLAS – 1989

Rua Conselheiro Nébias, 1384 (Campos Elísios)
Caixa Postal 7186 – Tel.: (011) 221-9144 (PABX)
01203 São Paulo (SP)

1. ed. 1988; 2. ed. 1989

ISBN 85-224-0398-8

Impresso no Brasil/**Printed in Brazil**

Depósito legal na Biblioteca Nacional conforme Decreto nº 1.825, de 20 de dezembro de 1907.

Capa: Paulo Ferreira Leite
Supervisão gráfica: Sergio Gerencer

Dados de Catalogação na Publicação (CIP) Internacional
(Câmara Brasileira do Livro, SP, Brasil)

Casari, Nélson, 1931-
C33d dBASE II PLUS para MSX : programável : curso prático / Nélson Casari. - - São Paulo : Atlas, 1988.

Bibliografia.
ISBN 85-224-0398-8

1. dBASE II PLUS (Programa de computador) 2. MSX (Computadores) – Programação I. Título.

CDD-001.6425
-001.642

88-1779

Índices para catálogo sistemático:

1. dBASE II PLUS : Computadores : Programas : Processamento de dados 001.6425
2. MSX : Computadores : Programas : Processamento de dados 001.642

SUMÁRIO

INTRODUÇÃO, pág. 13

1 - PROGRAMAÇÃO EM dBASE - CONCEITUAÇÃO, 15

- ARQUIVO DE COMANDOS, 16
- PROGRAMA, 16
- MODIFY COMMAND, 16

2 - CRIAÇÃO, EDIÇÃO E ALTERAÇÃO DE PROGRAMAS, 17

2.1 - CRIAÇÃO E EDIÇÃO DE PROGRAMA, 17
- Comando MODIFY COMMAND, 17
2.2 - ALTERAÇÃO OU CORREÇÃO DE PROGRAMA, 18
2.3 - EXECUÇÃO DE PROGRAMA EM dBASE, 19
- Comando DO, 19

3 - INICIAÇÃO EM PROGRAMAÇÃO dBASE - SAÍDA DE DADOS, 21

3.1 - CONTROLANDO O CURSOR E A TELA DE VÍDEO, 21
3.1.1 - POSICIONAMENTO DO CURSOR NA TELA DE VÍDEO E DO CABEÇOTE DA IMPRESSORA, 22
- Comando @, 22
3.1.2 - IMPRESSÃO NA TELA DE VIDEO E ATRAVÉS DA IMPRESSORA, 24
- Comando SAY, 24
- Comando SET FORMAT, 25

3.1.3 - OUTROS COMANDOS PARA UTILIZAÇÃO DA TELA DE VÍDEO E PARA IMPRESSÃO, 26
- Comandos DISPLAY e LIST, 27
- Comandos ? e ??, 27
- Comando ERASE, 28
- Comando SET TALK ON/OFF, 28
- Comandos TEXT e ENDTEXT, 28
3.1.4 - COMENTARIOS NO VÍDEO E/OU ATRAVES DE IMPRESSORA E EM PROGRAMAS, 29
- Comando NOTE, 30
- Comando *, 30
- Comando REMARK, 31
3.1.5 - SAÍDA DE DADOS, 31
3.2 - PROGRAMAÇÃO PARCIAL DE MENU DE OPÇÕES, 32

4 - FORMATAÇÃO DE DADOS EM ENTRADAS E SAíDAS, 39
- Cláusula PICTURE, 39
- Cláusula USING, 40

5 - ENTRADA DE DADOS ATRAVÉS DE PROGRAMAS, 42
- Comando GET, 43
- Comando READ, 44
- Comando CLEAR GETS, 46

6 - ALTERAÇÃO E ELIMINAÇÃO DE DADOS VIA PROGRAMAS, 53
6.1 - ALTERAÇÃO DE DADOS, 53
6.2 - ALTERAÇAO DE DADOS ATRAVÉS DOS COMANDOS GET E READ, 54
6.3 - ALTERAÇÃO DE DADOS ATRAVÉS DOS COMANDOS REPLACE, EDIT, BROWSE E CHANGE, 54
6.4 - ALTERAÇÃO DE DADOS ATRAVÉS DO COMANDO UPDATE, 58
6.5 - ELIMINAÇÃO DE DADOS, 59

7 - COMANDOS DE REPETIÇÃO (LAÇO) E DE DESVIO, 61
- Comando DO WHILE... ENDDO, 62
- Comando LOOP, 64
- Comando IF... ELSE... ENDIF, 65
- Comando DO CASE... OTHERWISE... ENDCASE, 68

7.1 - COMPLETANDO O MÓDULO DO MENU COM DUAS VERSÕES, 71

8 - MENSAGENS PROGRAMADAS, 77

8.1 - MENSAGEM COM TEMPO DE EXIBIÇÃO DEPENDENTE, 78
- Comando WAIT, 80
8.2 - MENSAGEM COM TEMPO DE EXIBIÇÃO DETERMINADO, 81

9 - RETORNO E CANCELAMENTO NA EXECUÇÃO DE PROGRAMA, 82

9.1 - RETORNO EM EXECUÇÃO DE PROGRAMA, 82
- Comando RETURN, 83
9.2 - CANCELAMENTO DE EXECUÇÃO DE PROGRAMA, 84
- Comando CANCEL, 84
9.3 - CANCELAMENTO DE EXECUÇÃO ATRAVÉS DE DO WHILE... ENDDO, 84

10 - ELABORAÇÃO DOS MÓDULOS COMPLEMENTARES
DO PROGRAMA "CONTROLE DE ESTOQUE", 86

10.1 - O USO DE FLUXOGRAMAS EM PROGRAMAÇÃO, 87
10.2 - ELABORAÇÃO DO MÓDULO OPC-1 - ENTRADA DE PRODUTO NOVO, 88
10.3 - ELABORAÇÃO DO MÓDULO
OPC-2 - ENTRADA DE PRODUTO NÃO NOVO, 92
10.4 - ELABORAÇÃO DO MÓDULO OPC-3 - SAÍDA DE PRODUTO, 95
10.5 - ELABORAÇÃO DO MÓDULO OPC-4 - DEVOLUÇÃO DE PRODUTO, 97
10.6 - ELABORAÇÃO DO MÓDULO OPC-5 - PESQUISA DE PRODUTO, 99
10.7 - ELABORAÇÃO DO MÓDULO OPC-6 - EMISSÃO DE LISTAGEM, 101
10.8 - FLUXOGRAMAS DO PROGRAMA "CONTROLE DE ESTOQUE", 108
- FLUXOGRAMA GERAL DO PROGRAMA "CONTROLE DE ESTOQUE", 109
- FLUXOGRAMA DO MÓDULO SUPERVISOR - MENU, 110
- FLUXOGRAMA DO MÓDULO OPC-1 - ENTRADA DE PRODUTO NOVO, 112
- FLUXOGRAMA DO MÓDULO OPC-2 -
ENTRADA DE PRODUTO NÃO NOVO, 114
- FLUXOGRAMA DO MÓDULO OPC-3 - SAÍDA DE PRODUTO, 116
- FLUXOGRAMA DO MÓDULO OPC-4 - DEVOLUÇÃO DE PRODUTO, 118
- FLUXOGRAMA DO MÓDULO OPC-5 - PESQUISA DE PRODUTO, 120
- FLUXOGRAMA DO MÓDULO OPC-6 - EMISSÃO DE LISTAGEM, 122
10.9 - MÓDULOS COMPLEMENTARES AO
PROGRAMA "CONTROLE DE ESTOQUE", 124
10.10 - ELABORAÇÃO DO MÓDULO OPC-ALT - ALTERAÇÃO DE PRODUTO, 125
10.11 - ELABORAÇÃO DO MÓDULO OPC-EDIT - EDIÇÃO DE PRODUTO, 127

10.12 - ELABORAÇÃO DO MÓDULO OPC-DEL - ELIMINAÇÃO DE PRODUTO, 129

11 - COMANDOS DA SÉRIE PLUS, 132

- Comando PLUS CDATE, 133
- PROGRAMA-TESTE "PLUS CDATE", 134
- Comando PLUS DDATE, 134
- PROGRAMA-TESTE "PLUS DDATE", 135
- Comando PLUS ADATE, 136
- PROGRAMA-TESTE "PLUS ADATE", 137
- Comando PLUS TDATE, 137
- PROGRAMA-TESTE "PLUS TDATE", 138
- Comando PLUS CHANGE, 139
- PROGRAMA-TESTE "PLUS CHANGE", 140
- Comando PLUS LZERO, 140
- PROGRAMA-TESTE "PLUS LZERO", 141
- Comando PLUS MOD11, 142
- PROGRAMA-TESTE "PLUS MOD11", 142
- Comando PLUS CGC, 143
- PROGRAMA-TESTE "PLUS CGC", 144
- Comando PLUS CPF, 144
- PROGRAMA-TESTE "PLUS CPF", 145
- Comando PLUS FORMAT, 145
- PROGRAMA-TESTE "PLUS FORMAT", 147
- Comando PLUS CODE, 147
- PROGRAMA-TESTE "PLUS CODE", 148

12 - RECURSOS SOFISTICADOS, 150

12.1 - INSERÇÃO OBRIGATÓRIA DE DATA, 151
- MÓDULO "OBRIGDAT", 152

12.2 - UNIFORMIZAÇÃO DE DATAS, 153
- MÓDULO "DTUNIFOR", 153

12.3 - CONVERSÃO E PADRONIZAÇÃO DE VALORES MONETÁRIOS, 155
- MÓDULO "PADR$BR", 156

12.4 - TROCA DE BRANCOS A ESQUERDA DE NÚMEROS POR ZEROS, 157
- MÓDULO "TROCZERO", 158

12.5 - MARCAÇÃO DE ENDDOs, ENDIFs, ENDCASEs, 159

12.6 - ATRIBUIÇÃO DE NÚMEROS OU VALORES ATRAVÉS DO COMANDO ACCEPT, 160

13 - DEPURAÇÃO DE PROGRAMAS, 163

- Comando SET ECHO ON/OFF, 164
- Comando SET STEP ON/OFF, 166
- Comando SET DEBUG ON/OFF, 168

14 - PROGRAMAÇÃO ESTRUTURADA E PROGRAMAÇÃO TOP DOWN, 170

14.1 - PROGRAMAÇÃO ESTRUTURADA, 170
- ESTRUTURA SEQUENCIAL OU DE SEQUÊNCIA SIMPLES, 171
- ESTRUTURA CONDICIONADA OU DE COMANDOS SELECIONADOS POR IF... ELSE... ENDIF OU DO CASE... OTHERWISE... ENDCASE, 171
- ESTRUTURA CONDICIONADA SIMPLES, 172
- ESTRUTURA CONDICIONADA COMPLEXA, 172
- ESTRUTURA DE "LAÇO" (LOOP) OU DE CICLO REPETITIVO, 173

14.2 - PROGRAMAÇÃO TOP DOWN, 173

ÍNDICE DOS COMANDOS ESTUDADOS NESTE LIVRO, 175

BIBLIOGRAFIA, 176

INTRODUÇÃO

Antes de ser consubstanciado neste livro, o MODO PROGRAMÁVEL do dBASE II PLUS foi inteiramente retalhado por nós. Todos os comandos e recursos disponíveis no mesmo foram minuciosamente estudados e experimentados, um a um, com a finalidade de serem descritos de maneira objetiva e com a maior clareza possível.

Como não podia deixar de ser, em se tratando de introduzir e comentar o modo programável do dBASE, no Capítulo 1 é feita uma breve explanação sobre criação, edição, alteração e execução de programas, com apresentação dos comandos próprios para tais fins.

A partir do segundo capítulo é iniciada a introdução gradativa dos demais comandos do modo programável, com exemplificações avulsas e em módulos de programa funcionais.

Visando facilitar a sua assimilação, os comandos foram agrupados segundo a sua finalidade, sendo introduzidos primeiramente os que são aplicados no controle do cursor do sistema, do cabeçote da impressora e da tela de vídeo, seguindo-se os próprios para formatação de dados em entradas e saídas, entrada, alteração e eliminação de dados, repetição, mensagens, retorno e cancelamento de programas etc.

Paralelamente com a introdução dos comandos e recursos básicos do modo programável do dBASE, um programa para controle de estoque é de-

senvolvido, com detalhamento de todos os módulos que o compõem.

Após o desenvolvimento completo do programa para controle de estoque, são introduzidos os comandos avançados e recursos sofisticados, com exemplificação e criação de módulos para aplicações diversas.

Procuramos apresentar neste livro o modo programável do dBASE no estilo de um Curso Prático, acreditando que, assim, seus comandos e recursos possam ser rápida e plenamente assimilados, possibilitando aos interessados, em curto espaço de tempo, aplicá-los profissionalmente de maneira ampla e irrestrita.

Com a publicação deste nosso segundo livro sobre o dBASE II PLUS esperamos estar colocando à disposição dos interessados em profissionalizar-se em programação dBASE as informações necessárias para que seu objetivo seja coroado de êxito.

São Paulo, julho de 1988.

O Autor.

1 - PROGRAMAÇÃO EM dBASE - CONCEITUAÇÃO

Como vimos no primeiro volume deste Curso, além dos comandos interativos, o dBASE dispõe de comandos especiais que constituem a sua linguagem de programação, a qual permite maximizar sua utilização no gerenciamento de bancos de dados.

Por possuir uma linguagem de programação própria que lhe confere amplos recursos, a par de uma linguagem de consulta potente e ágil, o dBASE se distingue dos demais sistemas de gerenciamento de bancos de dados, sendo praticamente imbatível em sua área.

Sabemos que, para cumprir determinadas tarefas, o microcomputador precisa receber comandos próprios do sistema que o controla. Tal sistema pode ser:

- o seu próprio - residente - complementado por uma linguagem de computação - BASIC, por exemplo -, como é o caso da maioria dos microcomputadores, inclusive os de padrão MSX.

ou:

- um sistema operacional independente - MSX DOS, HBDOS, por exemplo - através do qual outro sistema ou linguagem de computação - dBASE, por exemplo - passa a controlar o microcomputador.

Assim é que, através de um dos sistemas operacionais citados, o dBASE se torna ativo e disponível ao usuário, podendo ser usado no modo direto - ou interativo -, executando isolada e imediatamente cada comando próprio recebido, ou no modo indireto - ou programável -, executando lotes de comandos próprios de uma só vez.

Quando os comandos agrupados num lote são em pequeno número - até 10, por exemplo - e não incluem comandos condicionais ou de desvio, recebem o nome de

ARQUIVO DE COMANDOS.

Quando o agrupamento de comandos inclui quantidade considerável deles, e também de comandos condicionais ou de desvio, é denominado

PROGRAMA.

Do exposto, conclui-se que PROGRAMAR em dBASE consiste em agrupar adequadamente comandos próprios do SISTEMA dBASE, de modo que o microcomputador possa executá-los em lotes e seqüencialmente, com a finalidade de executar tarefas específicas previstas para tais comandos e suas combinações.

Para agrupar comandos do dBASE em ARQUIVO DE COMANDOS ou em PROGRAMA, utiliza-se o comando

MODIFY COMMAND.

MODIFY COMMAND já foi estudado no primeiro volume deste Curso, no item relativo a ARQUIVO DE COMANDOS. Será estudado novamente neste volume, com relação à PROGRAMAÇÃO em dBASE.

2 - CRIAÇÃO, EDIÇÃO E ALTERAÇÃO DE PROGRAMAS

2.1 - CRIAÇÃO E EDIÇÃO DE PROGRAMA

Vimos que programar em dBASE consiste em agrupar adequadamente, através de digitação, comandos próprios do sistema, de modo que possam ser executados em lotes e seqüencialmente.

Comando MODIFY COMMAND

O agrupamento de comandos do dBASE na forma de programa é feito através do comando MODIFY COMMAND, que ativa o editor do sistema, permitindo que os comandos digitados seqüencialmente, um por linha, sejam armazenados em arquivo próprio para execução em lote. MODIFY COMMAND é usado em conformidade com a sintaxe indicada a seguir:

MODIFY COMMAND <nome do arquivo>

Assim, a criação e a edição de programa em dBASE são feitas através de MODIFY COMMAND, com auxílio das teclas de controle próprias do comando (descritas no primeiro volume deste Curso e resumidas em tabelas de fácil acesso para consultas) e consistem em:

- acionar o comando MODIFY COMMAND <nome do arquivo>;

- digitar os comandos desejados com sua sintaxe correta, um por linha, seqüencialmente na ordem em que devem ser executados;

- encerrar a edição do programa com auxílio das teclas de controle CONTROL e W.

Como a sintaxe de MODIFY COMMAND requer que seja especificado o nome do arquivo em que será armazenado o programa, ao ser comandado, MODIFY COMMAND verifica se tal arquivo já existe no disquete alojado no "disk drive" corrente. Em caso negativo, exibe no vídeo a mensagem "Arq. novo" e, após limpar completamente a tela de vídeo, ativa o modo de edição FULL SCREEN do dBASE a fim de que sejam digitados os comandos que comporão o programa.

Depois de digitados e conferidos os comandos de um novo programa e encerrada sua edição através das teclas de controle CONTROL e W, ele é armazenado no arquivo aberto por MODIFY COMMAND e gravado no disquete automaticamente.

2.2 - ALTERAÇÃO OU CORREÇÃO DE PROGRAMA

Como dissemos, ao ser ativado, MODIFY COMMAND verifica automaticamente se existe no disquete alojado no "disk drive" corrente arquivo com o nome especificado. Em caso negativo, como vimos, abre um novo arquivo com o nome indicado. Em caso positivo, ou seja: existindo no disquete um arquivo com nome idêntico, exibe no vídeo o seu conteúdo, também no modo de edição FULL SCREEN, possibilitando efetuar alterações ou correções no mesmo.

Alterações e correções no programa editado no modo FULL SCREEN do dBASE são feitas também com auxílio das teclas de controle do comando MODIFY, as quais permitem deslocamento diversificado do cursor e inserção ou deleção de caracteres ou de linhas de programas, entre outros recursos. As teclas de controle do comando MODIFY e de outros são descritas no primeiro volume desde Curso e suas funções são resumidas em tabelas concisas e visuais que facilitam seu acesso e consultas.

Assim, alterações e correções em programas são feitas através de MODIFY COMMAND e consistem em:

- acionar o comando MODIFY COMMAND com o nome do arquivo do programa a ser trabalhado;
- fazer as alterações ou correções desejadas com auxílio das teclas de controle do comando MODIFY;
- encerrar a edição do programa com auxílio das teclas de controle CONTROL e W.

Resumindo: depois de gravado em disquete o arquivo que contém um programa, este poderá ser alterado ou corrigido a qualquer momento, bastando acionar MODIFY COMMAND com o seu nome.

Outro editor de texto que não o próprio pode ser usado para criar e editar programas em dBASE. Nos microcomputadores da linha MSX pode ser usado o WORDSTAR.

Em tal caso, todavia, o processo se torna mais complicado, já que a edição e a operação do programa dependerão de sistemas diferentes, obrigando o usuário a carregá-los alternativamente na memória da máquina, o que poderá ser muito trabalhoso se o programa demandar diversas correções e alterações.

2.3 - EXECUÇÃO DE PROGRAMA EM dBASE

Uma vez criado, elaborado e gravado em disquete, um programa em dBASE é executado através do comando indicado a seguir:

Comando DO

Tal comando, como vimos no item 19.2 do primeiro volume de nosso Curso, tem a seguinte sintaxe:

DO <nome do arquivo de comandos
ou do arquivo de programa>

Assim, dado tal comando através do dBASE ativado no microcomputador, o programa será prontamente executado, desde que esteja no disquete alojado no "disk drive" corrente.

3 – INICIAÇÃO EM PROGRAMAÇÃO dBASE – SAÍDA DE DADOS

Como vimos no primeiro volume deste Curso, no MODO INTERATIVO do dBASE o usuário apenas "comanda" ao dBASE o que fazer, sem preocupar-se com certos detalhes como, por exemplo, determinar a localização dos elementos exibidos na tela de vídeo, sendo isto feito automaticamente pelo sistema.

No MODO PROGRAMAVEL é o usuário quem deve controlar a tela de vídeo quanto à disposição dos elementos, em grande parte das execuções, como veremos. Por essa razão, a criação de telas para entrada e saída de dados torna-se um fator importante na programação em dBASE, do qual depende muito o bom desempenho do programa.

Em vista do exposto, serão descritos prioritariamente os comandos para utilização e controle da tela de vídeo que, além de ser o principal elo de ligação com o usuário, é também o principal veículo de saída de dados do sistema, vindo em seguida a impressora.

3.1 – CONTROLANDO O CURSOR E A TELA DE VÍDEO

Na versão do dBASE para 40 colunas, a tela de vídeo é dividida em 24 linhas e 40 colunas.

Na versão do dBASE para 80 colunas, a tela de vídeo é dividida em 24 linhas e 80 colunas.

As linhas são numeradas de 0 a 23, a partir da parte superior da tela de vídeo.

As colunas são numeradas de 0 a 39 na versão de 40 colunas e de 0 a 79 na versão de 80 colunas, a partir do lado esquerdo do vídeo.

Tanto na versão de 40 colunas como na de 80, a linha de número 0 é usada pelo dBASE como linha de "status", isto é, linha em que determinadas condições do sistema são informadas ao usuário. Por tal razão, convém não ser usada em programas.

3.1.1 - POSICIONAMENTO DO CURSOR NA TELA DE VÍDEO E DO CABEÇOTE DA IMPRESSORA

Comando @

*** O comando @ é utilizado em conjunto com o comando SAY para controlar o posicionamento do cursor na tela de vídeo e do cabeçote da impressora no papel de impressão, assinalando o ponto para início de impressão através de SAY. Sua sintaxe é:

```
@ x,y
```

sendo x a coordenada referente à linha e y a coordenada correspondente à coluna.

Para a tela de vídeo as coordenadas x e y podem assumir os valores indicados a seguir:

```
x = 0 a 23
y = 0 a 39 ou 79
```

Para a impressora, x e y podem assumir os valores:

x = 0 a n
y = 0 a 254

sendo n qualquer número.

Em se tratando de saídas para a impressora, é preciso lembrar que os valores de x e y devem ser sempre ascendentes - ou crescentes -, já que não é possível fazer o cabeçote da impressora retornar da direita para a esquerda para imprimir na mesma linha ou fazer o papel ou formulário contínuo retroceder para imprimir em linha de número menor ao da última impressa. Se o comando @ x,y a ser executado tiver um valor para x menor do que o determinado para x em comando idêntico anterior, por exemplo, provocará um avanço de página, ou de páginas, dependendo dos valores de x. O mesmo se verifica com relação a y.

Outro ponto que deve ser observado é com relação ao tipo de impressão utilizado na impressora. Por exemplo, se for usado o tipo de impressão normal numa impressora de 80 colunas, y poderá assumir o valor máximo 80. Se for utilizado o tipo de impressão condensada que imprime 16,7 caracteres por polegada linear, y poderá assumir o valor máximo 132. No caso de impressão condensada de 20 caracteres por polegada, o valor assumido por y poderá ser 160. Os valores indicados para y foram considerados em relação a uma impressora matricial de 80 colunas, tipo Mônica EI 6011, produzida por Elebra Informática S.A.

Com relação à quantidade de linhas por página, se for considerado o comprimento de formulários contínuos padronizados - 11 polegadas -, o número máximo de linhas que poderão ser impressas é de 66, com o entrelinhamento normal adotado pela maioria das impressoras, que é de 6 por polegada de altura.

Os valores de x e y podem ser representados por variáveis.

Quando usado apenas com as coordenadas, isto é, sem SAY, o comando @ x,y imprime "brancos" na linha especificada pela coordenada x, desde a coluna indicada na coordenada y até a última coluna da linha.

3.1.2 - IMPRESSÃO NA TELA DE VÍDEO E ATRAVÉS DA IMPRESSORA

Comando SAY

*** O comando SAY executa a impressão de dados (entendendo-se "dados" como conteúdos de variáveis ou de campos de registros), de expressão (entendendo-se "expressão" como uma expressão aritmética ou matemática ou o resultado de sua execução) e de texto (entendendo-se texto como qualquer conjunto de caracteres alfanuméricos indicados no comando entre aspas), na tela de vídeo ou através de impressora. Sua sintaxe é:

```
@ x,y SAY <campo, variável,
     expressão ou texto> [USING <"formato">]
```

SAY não é válido sem o comando @ x,y.

Exemplos com @ x,y e SAY:

```
@ 1,10 SAY "Linha 1, coluna 10 do vídeo."
@ 3,10 SAY "----------------------------"
@ 5,20 SAY 5*12/60+15
@ 7,15 SAY QT
```

QT no último exemplo representa o nome de uma variável de memória ou de campo de registro de arquivo.

Na condição normal de funcionamento do dBASE, o comando SAY gera impressão apenas na tela de vídeo. Para que imprima através de impressora é preciso que seja ativado o comando descrito a seguir:

Comando SET FORMAT

SET FORMAT é dotado de duas funções diferentes.

*** Uma das funções de SET FORMAT é direcionar a impressão de dados gerada através do comando @ x,y SAY para a tela de vídeo ou para a impressora. Nessa função é regido por duas sintaxes:

SET FORMAT TO PRINTER ou TO PRINT

SET FORMAT TO SCREEN

Ao ser carregado o dBASE no microcomputador, SET FORMAT é ativado automaticamente na condição TO SCREEN, sendo essa, pois, a sua condição "default".

A principal utilização de SET FORMAT é na emissão de relatórios no modo programável do dBASE através de impressora, quando, por força do uso praticamente inevitável do comando @ x,y SAY, deve ser acionado na condição TO PRINTER. Depois de ativado nessa condição, para redirecionar as saídas de @ x,y SAY para o vídeo, SET FORMAT deve ser acionado na condição TO SCREEN.

*** Outra função de SET FORMAT é permitir o uso de telas personalizadas para entrada e edição de dados no MODO INTERATIVO do dBASE, através dos comandos APPEND, EDIT e INSERT. Nessa função é regido pela seguinte sintaxe:

SET FORMAT TO <nome do arquivo>

Como sabemos, no MODO INTERATIVO, o dBASE utiliza suas próprias telas padronizadas quando tais comandos são executados. Mas as telas padronizadas apresentam algumas inconveniências como, por exemplo, não descrever a finalidade e natureza dos campos de registros e não permitir que sejam selecionados para entrada via digitação apenas determinados campos, já que outros poderão ter seus conteúdos definidos indiretamente através de comandos diversos.

Outro fator importante em matéria de entrada e edição de dados é o estético, que pode tornar a sua manipulação mais ou menos eficiente, mais ou menos cansativa, mais ou menos otimizada.

Uma tela personalizada é criada e armazenada em arquivo próprio através do comando MODIFY COMMAND do mesmo modo que é criado e armazenado um arquivo de comandos ou um programa ou, ainda, um módulo deste, com a diferença única de que o nome do arquivo deve receber a extensão FMT, para ser classificado e processado pelo dBASE como um ARQUIVO DE FORMATO e ser usado como tal.

A edição de um ARQUIVO DE FORMATO é feita com os comandos normais para utilização da tela de vídeo, como @ x,y SAY, ?, GET, CLEAR GETS, READ, ERASE etc., que estão sendo estudados neste capítulo.

A edição de um arquivo de formato é encerrada normalmente através das teclas de controle CONTROL e W, pressionadas simultaneamente.

Depois de criado e editado, um ARQUIVO DE FORMATO é acionado para uso após já estar em uso o arquivo de dados (.DBF) ao qual se relaciona. Exemplo:

USE CTRLSTOQ <RETURN>
SET FORMAT TO <nome do arquivo> <RETURN>

Nessa situação, todas as vezes em que forem acionados os comandos APPEND, EDIT e INSERT, o dBASE utilizará a tela personalizada para entrada e edição de dados, em vez de suas telas padronizadas.

Para reverter o processamento à condição normal, deve ser dado o comando:

SET FORMAT TO SCREEN <RETURN>

3.1.3 - OUTROS COMANDOS PARA UTILIZAÇÃO DA TELA DE VÍDEO E PARA IMPRESSÃO

Comandos DISPLAY e LIST

Os comandos DISPLAY e LIST foram descritos e estudados no primeiro volume deste Curso. Suas aplicações no modo programável do dBASE são válidas e produzem os mesmos resultados que no modo direto. Todavia, sua utilização no modo programável é bastante reduzida, restringindo-se a casos especiais.

Comandos ? e ??

Os comandos ? e ?? são bastante usados em programas dBASE para reproduzir conteúdos de variáveis ou de campos de registros, comentários, expressões etc., seja na tela de vídeo ou através de impressora.

Suas regras de aplicação são as mesmas descritas para o modo interativo. Vamos relembrá-las:

? imprime sempre na primeira linha disponível o que estiver especificado no comando. Se nada for especificado, imprimirá uma linha inteira de "brancos".

?? imprime em seguida à impressão do comando ? precedente, isto é, inicia a impressão na mesma linha produzida por ?, a partir da primeira coluna disponível.

Não é possível determinar diretamente no comando ? a linha e a coluna para início de impressão.

Indiretamente, a linha para início de impressão através de ? pode ser determinada com auxílio do comando @ x,y, já que, usado imediatamente após uma linha "em branco" impressa por tal comando, ? iniciará a impressão a partir dessa linha.

A coluna para início de impressão poderá ser determinada também de modo indireto, imprimindo-se "brancos" antes da coluna visada.

O uso de vírgulas permite que o comando ? seja válido para imprimir na mesma linha diversos elementos diferentes, como texto, conteúdo de variáveis ou de campos de registros, expressões, dados etc.

Exemplos:

```
? "        Digite os valores usando ponto e não"
? "              vírgula antes das casas decimais."
? "O custo total é:",PRV * QT
? "     ",STOQ,"     ",VALOR
```

(PRV, QT, STOQ e VALOR são nomes de variáveis ou campos em uso.)

Comando ERASE

*** O comando ERASE esvazia ou "limpa" a tela de vídeo.

ERASE já foi descrito no primeiro volume deste Curso. É um comando bastante utilizado no modo programável do dBASE.

Comando SET TALK ON/OFF

SET TALK na condição ON caracteriza o MODO INTERATIVO do dBASE, que exibe resposta na tela de vídeo para a maioria dos comandos executados nesse modo.

Ao ser ativado o dBASE, SET TALK entra automaticamente em ON. Em programas geralmente é colocado na condição OFF, para que a exibição de resultados de comandos seja feita de acordo com a conveniência do programa e do usuário e através de comandos específicos.

SET TALK OFF e ERASE geralmente são os dois comandos iniciais em qualquer programa em dBASE II.

Comandos TEXT e ENDTEXT

TEXT e ENDTEXT são comandos muito úteis para uso da tela de vídeo e da impressora no modo programável do dBASE, para impressão de textos em geral, incluindo comentários, lembretes ao usuário, menus etc.

O texto intercalado entre TEXT e ENDTEXT é reproduzido na tela de

vídeo ou através de impressora exatamente na mesma disposição como foi digitado, fato esse que facilita deveras a criação de menus, por exemplo, ou de textos com formatos especiais.

Sua sintaxe é:

```
TEXT
<menu, texto, comentários etc.>
ENDTEXT
```

Exemplos:

```
TEXT

        1. ENTRADA DE DADOS
        2. ATUALIZAÇÃO DE DADOS
        3. CANCELAMENTO DE DADOS
        4. ENCERRAMENTO

           < ESCOLHA A OPÇÃO >

ENDTEXT

TEXT
ATENÇÃO! Ao usar arquivos de dados,
         "backups" devem ser feitos
         em intervalos adequados, a
         título de prevenção contra
         acidentes (falta de luz ou
         defeito no equipamento.)

ENDTEXT
```

3.1.4 - COMENTÁRIOS NO VÍDEO E/OU ATRAVÉS DE IMPRESSORA E EM PROGRAMAS

Em certos casos, a impressão de comentários, vinhetas, títulos,

linhas de demarcação etc. em telas de vídeo ou em relatórios em papel pode ser conveniente, por diversas razões, inclusive as de ordem estética. Pode mesmo constituir uma necessidade em determinadas modalidades de programas.

Por outro lado, em função da complexidade do programa, a introdução de comentários elucidativos em sua listagem pode ser uma necessidade, a fim de assinalar para o programador pontos críticos carentes de revisão ou para os quais deverá retornar atenção futura. Comentários são também introduzidos em programas para elucidar sua finalidade ou as funções de determinados trechos do mesmo. Obviamente, porém, comentários desta modalidade não devem ser exibidos no vídeo nem ocasionar interrupção do programa ou prejudicar o seu desempenho.

Os comandos descritos a seguir são próprios para as finalidades ora comentadas:

Comando NOTE

*** NOTE permite introduzir em programas comentários elucidativos não executáveis e que não alteram o fluxo de execução dos mesmos. Não são exibidos no vídeo nem reproduzidos através de impressora. Sua sintaxe é:

```
NOTE <comentários>
```

Exemplo:

```
NOTE - Módulo para entrada de dados.
```

Comando *

*** O mesmo que NOTE. Não confundir com a FUNÇÃO *, estudada no primeiro volume deste Curso.

Comando REMARK

*** REMARK gera no vídeo ou através da impressora os dizeres e/ou caracteres que especifica. Sua sintaxe é:

```
REMARK <dizeres, caracteres ou expressão>
```

Enquanto NOTE e * servem para introduzir em programas comentários elucidativos que não são exibidos no vídeo nem reproduzidos através de impressora, sendo úteis apenas para exame ou consulta através da listagem do programa, REMARK possibilita a exibição de mensagens, lembretes, vinhetas, títulos etc. através do vídeo e/ou sua reprodução através de impressora.

Exemplos:

```
REMARK ***** TELA DE DIGITAÇÃO *****
REMARK ################################
REMARK Digite os dados em caixa alta
```

3.1.5 - SAÍDA DE DADOS

Com base no exposto até este ponto, podemos dizer que, ao mesmo tempo que descrevemos os diversos comandos e recursos disponíveis no dBASE II PLUS para controle do cursor do sistema, do cabeçote da impressora e da tela de vídeo, expusemos também os meios de SAÍDA DE DADOS através da tela de vídeo e da impressora, já que a liberação de dados para as mesmas depende quase que inteiramente daqueles elementos para efetivar-se, sendo fácil perceber a estreita relação existente entre tais comandos e recursos e a saída de dados.

De certa forma também o processo de ENTRADA DE DADOS foi parcialmente abordado, já que o mesmo se efetiva valendo-se inteiramente da tela de vídeo quando as entradas são feitas através do teclado.

Evidentemente, porém, ambos os processos dependem da complementação de diversos outros comandos e recursos para apresentar desempenho

otimizado. Tais comandos e recursos serão gradativamente introduzidos nos capítulos seguintes.

3.2 - PROGRAMAÇÃO PARCIAL DE MENU DE OPÇÕES

Baseados no que foi estudado até esta parte, elaboraremos um módulo de programa gerador de "menu de opções" que será utilizado como módulo supervisor ou "driver" de um sistema de "CONTROLE DE ESTOQUE" que desenvolveremos a partir dele. Inicialmente faremos apenas a parte que é responsável pela geração de sua imagem na tela de vídeo.

Na modalidade TOP DOW de programação estruturada, que é o modelo de programação em que se enquadra o modo programável do dBASE, o "menu de opções" é o primeiro módulo criado para um programa, sendo os demais criados em função dele. O módulo do menu funciona qual um "roteiro" para elaboração geral do programa, durante cuja execução não só controla seus diversos módulos como também interage na sua integração.

A programação estruturada será discutida com mais detalhes no Capítulo 14. Por ora, cumpre-nos elaborar o programa gerador da tela do menu, com a finalidade de dar os primeiros passos em programação do sistema dBASE e demonstrar a aplicação de comandos já estudados e que são válidos para controle e utilização da tela de vídeo.

Dado o fato já comentado anteriormente de que a tela de vídeo é tratada pelo dBASE com 24 linhas e 40 ou 80 colunas, representando 960 ou 1920 pontos de referência para localização do cursor do sistema ou para exibição de caracteres na mesma, aconselha-se que a programação para seu uso adequado e estético seja baseada em "mapeamento" da distribuição dos elementos que deverão ser exibidos, sem o que os resultados a obter dependerão de tentativas, que poderão ser muitas e cansativas, ou de prática consumada.

Tal mapeamento poderá ser feito em formulário próprio para a finalidade ou em papel quadriculado comum, no qual os limites correspondentes à tela de vídeo e os números de linhas e colunas sejam marcados por qualquer meio.

MAPA DE TELA DE VÍDEO – 24 LINHAS X 40 COLUNAS

REFERÊNCIA: PROGRAMA "CONTROLE DE ESTOQUE" - MENU

MAPA DE TELA DE VÍDEO – 24 LINHAS X 40 COLUNAS

REFERÊNCIA:

Figura 3.2-A

Nosso programa gerador da exibição do menu utilizará a tela de vídeo de 40 colunas e será baseado no mapeamento ou "layout" reproduzido na Figura 3.2-A.

Como foi explicado, um programa em dBASE é iniciado com o acionamento do comando MODIFY COMMAND seguido do nome do arquivo que o armazenará. Assim, estando o dBASE ativado, comandamos:

MODIFY COMMAND MENU <RETURN>

e, após a tela de vídeo ser esvaziada e aparecer o cursor no canto esquerdo superior da mesma, podemos digitar as linhas de programa indicadas abaixo. Após a digitação de cada linha poderá ser pressionada a tecla RETURN para deslocar o cursor para o início da linha seguinte ou poderão ser pressionadas simultaneamente as teclas CONTROL e X, além da barra espaçadora. Correções, inserções e anulações de caracteres ou de linhas poderão ser feitas com auxílio das outras teclas de controle do comando MODIFY. No caso de dúvidas, poderão ser consultadas as tabelas próprias no primeiro volume desde Curso.

Linhas do programa a digitar:

```
* PROGRAMA "CONTROLE DE ESTOQUE"
* MÓDULO MENU - GERADOR DO MENU DE OPÇÕES
SET TALK OFF
ERASE
@ 5,10 SAY "CONTROLE DE ESTOQUE"
@ 6,10 SAY "-------------------"
@ 9.5 SAY "1 - ENTRADA DE PRODUTO NOVO"
@ 10,5 SAY "2 - ENTRADA DE PRODUTO NÃO NOVO"
@ 11,5 SAY "3 - SAÍDA DE PRODUTO"
@ 12,5 SAY "4 - DEVOLUCÃO DE PRODUTO"
@ 13,5 SAY "5 - PESQUISA DE PRODUTO"
@ 14,5 SAY "6 - EMISSÃO DE LISTAGEM"
@ 15,5 SAY "7 - ENCERRA O PROCESSAMENTO"
@ 17,5 SAY "<<< Digite o número da opção >>>"
```

Ao terminar a digitação, pressionar simultaneamente as teclas de controle CONTROL e W para findar a edição do programa e gerar sua gra-

vação automática no disquete, que deverá estar alojado no "disk drive" corrente para essa finalidade. O arquivo que o armazenará será gravado com o nome especificado pelo comando MODIFY COMMAND e receberá, também automaticamente, a extensão .CMD se outra não lhe tiver sido destinada pelo usuário quando digitou o comando. Conforme foi observado no primeiro volume deste Curso, quando os arquivos dBASE são gravados com as extensões próprias do sistema, elas não precisam ser incluídas nos comandos dados para seu uso ou execução, ao passo que, tendo extensões diferentes, escolhidas pelo usuário, estas terão de ser incluídas nos comandos dados para seu uso ou execução.

Depois de gravado no disquete, o programa poderá ser executado. Para tanto, deverá ser dado o comando indicado a seguir:

DO MENU <RETURN>

Sua execução produzirá a imagem de vídeo reproduzida a seguir:

```
        CONTROLE DE ESTOQUE
        -------------------


  1 - ENTRADA DE PRODUTO NOVO
  2 - ENTRADA DE PRODUTO NÃO NOVO
  3 - SAÍDA DE PRODUTO
  4 - DEVOLUÇÃO DE PRODUTO
  5 - PESQUISA DE PRODUTO
  6 - EMISSÃO DE LISTAGEM
  7 - ENCERRA O PROCESSAMENTO

  <<< Digite o número da opção >>>
```

Figura 3.2-B

O leitor deverá observar que o ponto de prontidão do dBASE retorna automaticamente ao vídeo após a exibição do menu na tela de vídeo, fato esse que não deverá ocorrer quando se tratar de um menu de opções

definitivo. Ocorre neste caso porque seu programa gerador ainda está incompleto em razão de, por ora, destinar-se apenas a gerar sua imagem na tela de vídeo. Normalmente, a função de um menu efetivo só pode ser encerrada através da escolha de opção de encerramento, condição que o nosso menu terá depois de acrescentarmos a seu programa gerador comandos ainda não estudados.

Veremos agora que a mesma imagem reproduzida na Figura 3.2-B pode ser gerada por apenas dois comandos, TEXT e ENDTEXT, em vez dos diversos @ x,y SAY, conforme ficou demonstrado através de seu programa gerador. O processo para uso de TEXT e ENDTEXT é descrito a seguir:

Estando o dBASE ativado no microcomputador, comandar:

MODIFY COMMAND MENU2 <RETURN>

Depois de a tela de vídeo estar limpa e o cursor estar posicionado no canto esquerdo superior, indicando prontidão para digitação, digitar as linhas de programa indicadas a seguir com auxílio das teclas de controle próprias do comando MODIFY:

```
* PROGRAMA "CONTROLE DE ESTOQUE"
* MÓDULO MENU2 - GERADOR DO MENU DE OPÇÕES
SET TALK OFF
ERASE
TEXT
```

Em seguida, deslocar o cursor para a posição desejada para o título do menu e digitá-lo no formato em que deverá ser exibido depois:

```
CONTROLE DE ESTOQUE
```

Continuando, digitar os demais elementos componentes do menu exatamente na posição e no formato desejados para sua reprodução:

```
--------------------------------


1 - ENTRADA DE PRODUTO NOVO
```

2 - ENTRADA ...
3.- ...

até o último elemento componente do menu. (Notar que as aspas não são necessárias quando são usados os comandos TEXT e ENDTEXT.)

Após terminar a composição (ou esboço) da imagem do menu na tela de vídeo, "pular" algumas linhas em branco e digitar:

ENDTEXT

Finalmente, para encerrar a edição do programa e produzir a sua gravação automática em disquete, teclar simultaneamente CONTROL e W.

Em seguida, executar o programa com o comando:

DO MENU2 <RETURN>

e observar a imagem gerada no vídeo.

Se na primeira tentativa a imagem gerada não for a desejada quanto à posição e ao formato dos elementos do menu, reeditar o programa com o comando:

MODIFY COMMAND MENU2 <RETURN>

e efetuar as correções desejadas com auxílio das teclas de controle, conforme foi explicado no item 2.2.

Comparando-se os dois programas para geração do mesmo menu, concluir-se-á que o segundo é bastante mais simples, exigindo menos trabalho de digitação e facilitando a estimativa de visualização da tela final que será obtida, já que sua imagem corresponderá à da digitação feita entre os comandos TEXT e ENDTEXT.

Além de facilitar a criação de menus, os comandos TEXT e ENDTEXT são recomendados especialmente para geração de textos explicativos ou de mensagens que devem ser exibidos durante a execução de um programa.

Como explicamos em relação ao módulo gerador de menu anterior, também este ainda não é definitivo, pois está incompleto. Deverão ser-lhe acrescentados comandos que serão estudados nos próximos capítulos.

As gravações já feitas dos dois diferentes módulos elaborados deverão ser preservadas, pois serão utilizadas logo mais quando completarmos o menu, cujo programa gerador terá também duas versões.

4 – FORMATAÇÃO DE DADOS EM ENTRADAS E SAÍDAS

Dados ou expressões podem ser formatados no dBASE, em entradas ou saídas, através das cláusulas PICTURE e USING, respectivamente.

Cláusula PICTURE

A cláusula PICTURE é usada exclusivamente com o comando GET na entrada de dados. O comando GET será estudado no capítulo seguinte, o qual trata da entrada de dados através de programas em dBASE II, mas a cláusula PICTURE está sendo introduzida com precedência a fim de que a descrição e a exemplificação do uso de GET possam ser feitas sem interrupção e amplamente, incluindo a utilização da cláusula em questão, já que GET e PICTURE são usados quase que invariavelmente em conjunto.

Uma das funções de PICTURE é formatar os dados ou expressões destinados a variáveis de memória e a campos de registros. A outra função sua é selecionar por restrição os dados que serão atribuídos a variáveis e a campos de registros. Ambas as funções são simultâneas.

O controle de formatação e a seleção de tipos são exercidos por PICTURE através de um gabarito (ou "máscara") criado pelo usuário com auxílio dos caracteres:

"" , / - . # 9 A X !

Funções dos caracteres:

"" determinam o tamanho do gabarito. Não são reproduzidos com os dados digitados para entrada, obviamente.

, / - ou outros separadores determinam separações nos dados ou expressões em conformidade com suas posições no gabarito. São reproduzidos com os dados digitados.

. separa as casas decimais do número. É repetido com os dados.

e 9 permitem a entrada apenas de números. Se os dados digitados se destinarem a variável ou campo numéricos, somente o ponto será válido como separador e, neste caso, só pode ser incluído no gabarito como separador de casas decimais. Vírgulas ou quaisquer outros separadores não podem ser incluídos no gabarito. Se a entrada se destinar a variável ou campo alfanuméricos, serão aceitos como separadores quaisquer caracteres que não sejam letras.

A permite a entrada de caracteres alfabéticos apenas (letras).

X permite a entrada de quaisquer caracteres alfanuméricos quando os mesmos se destinam a variável alfanumérica ou campo de caracteres. Quando a entrada se destina a variável ou campo de registro numéricos, permite apenas a entrada de números, aceitando somente o ponto (.) como separador decimal.

! permite a entrada de quaisquer caracteres, convertendo os alfabéticos minúsculos em maiúsculos.

Cláusula USING

A cláusula USING tem função similar à de PICTURE, mas é usada exclusivamente com o comando SAY, estudado no capítulo anterior. USING controla a formatação e os tipos de dados ou expressões em saídas para a tela de vídeo ou para a impressora.

A cláusula USING não é válida com outros comandos.

O controle de formatação e a seleção de tipos para saída de dados são feitos por USING com auxílio de um gabarito (ou "máscara") criado

pelo usuário com utilização dos seguintes caracteres:

"" / - . , # 9 A X ! $ *

Funções dos caracteres:

"" determinam o tamanho do gabarito. Não são reproduzidos com os dados, obviamente.

/ e - determinam a separação dos dados em conformidade com suas posições no gabarito. São reproduzidos juntamente com os dados.

. separa as casas decimais do número. É repetido com os dados.

, separa dígitos à esquerda do ponto decimal. É reproduzido juntamente com os dados.

e 9 permitem saída apenas de números. Se os dados contidos em variável ou campo de registro forem alfanuméricos, darão saída apenas a "espaços", isto é, sairão "brancos" nos lugares das letras. No gabarito formado com os caracteres # e 9 são válidos o ponto (.), espaço e outros separadores. O ponto separa dígitos decimais, enquanto que os outros separadores (vírgula, espaço etc.) são úteis para separar casas na parte inteira do número.

A permite a saída de caracteres alfabéticos apenas (letras). Se os dados forem constituídos por caracteres numéricos, eles serão substituídos por caracteres A.

X permite a saída de quaisquer caracteres, desde que não constituam valor aritmético. Caso contrário, dará saída a si mesmo.

! converte as letras minúsculas dos dados em letras maiúsculas.

$ ou * um ou outro será impresso nos lugares de zeros à esquerda do número, em conformidade com sua posição no gabarito. Se incluídos no gabarito nas posições das casas decimais, substituirão os seus dígitos na saída, ainda que os mesmos sejam maiores do que zero.

No capítulo seguinte será demonstrado amplamente o uso das cláusulas PICTURE e USING, de modo que deixamos de fazê-lo neste.

5 — ENTRADA DE DADOS ATRAVÉS DE PROGRAMAS

No modo programável do dBASE as entradas de dados via teclado são feitas quase que invariavelmente através do uso conjunto dos comandos GET e READ, embora possam ser feitas também por meio dos comandos APPEND e INSERT, usados mais apropriadamente no modo interativo e estudados no primeiro volume deste Curso.

Por oportuno, lembramos que todos os comandos tratados como próprios do modo interativo são igualmente válidos para uso no modo programável, devendo ser usados nesta modalidade sempre que possam produzir resultados otimizados ou os desejados pelo programador.

Em verdade, muitos dos comandos interativos são indispensáveis na elaboração de programas em dBASE e a prática tem-nos demonstrado que a melhor programação é obtida através do uso irrestrito dos comandos de seus dois modos de operação.

Assim, não há nenhum impedimento, por exemplo, em elaborar programas em que as entradas de dados sejam processadas por comandos como APPEND e INSERT, e a sua adoção ou não deve ser determinada por fatores peculiares a cada situação ou por conveniência do usuário.

Os comandos GET e READ são usados preferencialmente para entradas de dados no modo programável porque as suas características especiais permitem formatar a tela de vídeo adequadamente e "a gosto" para a fi-

nalidade, ao mesmo tempo que oferecem facilidades para a digitação dos dados e sua destinação a variáveis de memória e a campos de registros.

Comando GET

*** GET exibe ou imprime no vídeo, em local determinado pelo comando @ x,y, o conteúdo de variável predefinida ou de campo de registro de arquivo em uso. É usado obrigatoriamente com o comando @ x,y, podendo ser precedido opcionalmente pelo comando SAY.

Sua sintaxe é:

```
@ x,y [SAY <texto ou comentário>]
        GET <nome de variável ou de campo
             de registro> [PICTURE <"formato">]
```

Exemplos do uso de GET:

```
1. @ 5,15 GET A
2. @ 7,5 SAY "O conteúdo da variável é:" GET X
3. @ 9,10 SAY "O conteúdo do campo é:" GET NM
4. @ 11,15 SAY 1234567 USING "#,###,###.##"
5. @ 14,20 SAY "01.04.88" USING "XX/XX/XX"
```

Assumindo que o dBASE esteja ativado e que as variáveis A e X tenham sido previamente definidas e que o arquivo a que pertence o campo NM esteja em uso no momento de execução do comando, os exemplos acima produzirão:

1. Exibição do conteúdo da variável A na quinta linha e décima quinta coluna da tela de vídeo.
2. Exibição do conteúdo da variável X, após a frase impressa por SAY, na posição determinada pelas coordenadas x e y.
3. Exibição do conteúdo do campo de registro NM, após o comentário expresso por SAY, na posição determinada pelas coordenadas x e y.
4. Exibição de 1,234,567.00 na linha 11, coluna 15.
5. Exibição de 01/04/88 na linha 14, coluna 20.

Convém salientar que GET não cria a variável ou o campo de registro que enuncia. Se especificar variável não definida previamente ou campo de registro inexistente ou de arquivo fora de uso no momento de sua execução, ocasionará a interrupção do programa por erro de sintaxe e gerará a emissão de mensagem correspondente.

Se a variável enunciada por GET for numérica e tiver conteúdo nulo (0), será exibida na tela de vídeo a área que lhe é destinada normalmente pelo sistema, correspondente a 10 dígitos. Tal área será exibida com o sinal de dois pontos (:) como delimitadores extremos e um 0 ocupando a última posição. Se a variável tiver conteúdo não nulo, este será exibido em condição idêntica à descrita para conteúdo nulo.

Se a variável enunciada por GET for alfanumérica e não tiver conteúdo, será exibida no vídeo a área que lhe tiver sido destinada através do comando STORE, sendo tal área também demarcada nos extremos com o sinal de dois pontos (:). Se a variável já tiver conteúdo, este será exibido em condição idêntica à descrita para conteúdo não existente.

Se GET enunciar campo de registro, será exibida no vídeo a área que lhe corresponder, em conformidade com a estrutura do arquivo a que pertence, com o conteúdo que tiver, seja ele constituído por número ou caracteres alfanuméricos, seja apenas por espaços ("brancos").

Comando READ

*** READ interrompe o andamento do programa ao ser executado e desloca o cursor para a área correspondente à exibição feita pelo comando GET de conteúdo de variável ou campo de registro, permitindo que nessa mesma área seja digitado novo conteúdo ou que seja confirmado o já existente (o exibido). Aquele que permanecer será atribuído automaticamente ao mesmo campo de registro ou à mesma variável - ao que tiver sido enunciado - após pressionar RETURN. O conteúdo referido poderá ser qualquer: nulo ou não, caracteres ou simplesmente espaços ("brancos"). A sintaxe de READ é simplesmente:

READ

READ é válido apenas após a execução de um comando GET do tipo:

```
@ x,y [SAY <"comentário">] GET <variável/campo>
```

Exemplo de um pequeno módulo com @, SAY, GET, PICTURE e READ:

```
...
ERASE
STORE "        " TO DT
@ 5,15 SAY "Digite a data" GET
                         DT PICTURE "99/99/99"
READ
...
```

A primeira linha do módulo provoca o esvaziamento da tela.

A segunda linha define a variável DT e lhe atribui 8 "brancos".

A terceira linha produz no vídeo (na linha 5, a partir da coluna 15) a impressão do texto determinado em SAY, seguido do gabarito especificado em PICTURE, conforme reprodução feita a seguir:

```
Digite a data :  /  /  :
```

Finalmente, o comando READ (quarta linha do módulo exemplificado) posiciona o cursor no primeiro espaço em branco do gabarito, a fim de que sejam digitados os dados que serão atribuídos à variável DT. Como o gabarito foi criado com o caractere-chave 9, apenas números serão aceitos através da digitação, sendo impossível digitar letras ou outros caracteres. É interessante observar que, durante a digitação, o cursor saltará automaticamente os separadores, devendo o usuário estar atento ao fato, a fim de não digitar os dados erroneamente.

Quando READ é usado em conjunto com o comando GET, após a execução de ambos o cursor do sistema é deslocado automaticamente para a área em que é exibido o conteúdo da variável ou do campo de registro especificado em GET, permitindo digitar dados nessa mesma área, que é

operada pelo dBASE na modalidade "full screen" (= tela plena). Em tal condição, os dados digitados são destinados ou atribuídos direta e automaticamente à variável ou ao campo de registro enunciado no comando GET e cuja área foi exibida no vídeo, com ou sem conteúdo.

Depreende-se do exposto que:

- usado sem READ, GET apenas exibe no vídeo o conteúdo de variáveis de memória e de campos de registros predefinidos - ou já existentes - e disponíveis no momento de sua execução;
- usado em conjunto com READ, GET também exibe na tela de vídeo os conteúdos de variáveis e de campos de registros predefinidos e disponíveis no momento de sua execução, mas possibilitando, nessa condição, alterar tais conteúdos ou criá-los caso sejam nulos ou apenas espaços ("brancos").

Para atribuir números, valores ou conteúdos alfanuméricos a novas variáveis através do uso conjunto de GET e READ, as mesmas deverão ser previamente definidas ou criadas nas seguintes condições:

- com valor 0 (zero) no caso de variáveis numéricas;
- com espaços ("brancos") no caso de variáveis alfanuméricas.

Para atribuir números, valores ou conteúdos alfanuméricos a campos de registro novo de arquivo através do uso conjunto de GET e READ, o arquivo tem de estar em uso no momento de execução de ambos os comandos, e o novo registro - com campos "em branco", portanto - deverá ser requerido previamente no programa através do comando APPEND BLANK.

No capítulo seguinte, que tratará dos recursos do dBASE disponíveis para efetuar alterações ou atualizações de dados através de programas, será examinada a competência de GET e READ a esse respeito.

Comando CLEAR GETS

*** O comando CLEAR GETS evita que as áreas correspondentes a conteúdos de variáveis ou de campos de registros exibidos no vídeo através de GET sejam abrangidas por um comando READ subseqüente.

A sua sintaxe é simplesmente:

```
CLEAR GETS
```

Exemplo do uso de CLEAR GETS num pequeno módulo:

```
...
ERASE
@ 3,5 SAY "Código do produto" GET
                                COD PICTURE "##-###"
@ 5,5 SAY "Nome do produto" GET NM
CLEAR GETS
@ 7,10 SAY "Quantidade" GET QT PICTURE "9999"
@ 9,10 SAY "Preço" GET PRV PICTURE "####.##"
READ
...
```

Ao ser executado esse trecho do programa, o código e nome do produto serão exibidos normalmente no vídeo e, por força do comando CLEAR GETS, não serão alcançados pelo comando READ. Quando este for executado, o cursor será deslocado automaticamente para o campo de digitação criado por GET para a variável QT, prosseguindo depois para o campo de digitação correspondente a PRV e assim por diante.

Vamos agora elaborar um módulo de programa com o propósito de demonstrar efetivamente o uso de GET, READ e CLEAR GETS, já que tais comandos costumam causar a principiantes em programação em dBASE algum embaraço na sua assimilação, em vista de sua versatilidade e - por que não dizer? - de sua complexidade.

Vamos chamar esse módulo de PEGDADO. Assim, estando o dBASE ativado, comandar:

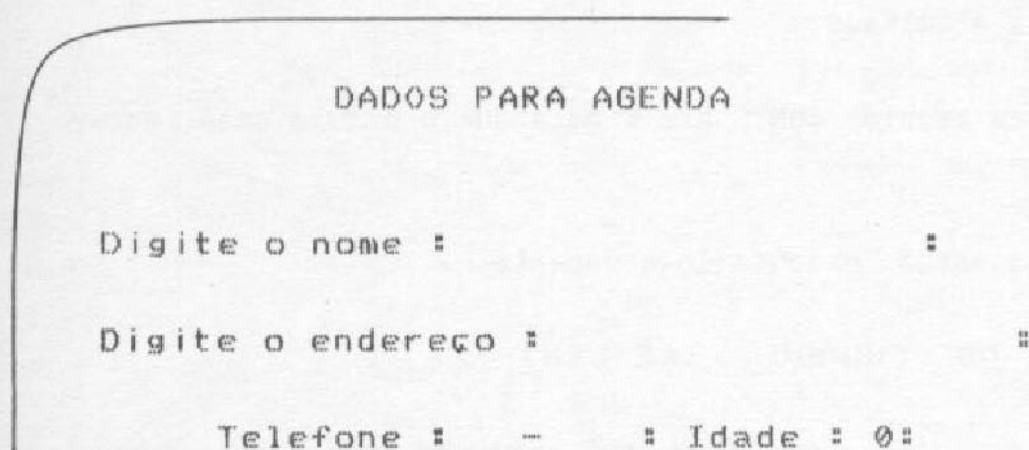
```
MODIFY COMMAND PEGDADO
```

e, em seguida, digitar:

```
SET TALK OFF
ERASE
STORE "                              " TO NOME
STORE "                              " TO END
STORE "          " TO TEL
STORE 0 TO IDADE
@ 3,10 SAY "DADOS PARA AGENDA"
@ 6,0 SAY "Digite o nome" GET NOME
@ 8,0 SAY "Digite o endereço" GET END
@ 10,5 SAY "Telefone" GET TEL PICTURE "999-9999"
@ 10,25 SAY "Idade" GET IDADE PICTURE "99"
READ
@ 14,0
? "NOME:",NOME
? "ENDEREÇO:",END
? "TEL.:",TEL," IDADE:",IDADE
```

Lembramos que no Capítulo 22 do primeiro volume deste Curso foi feito o estudo completo do uso de variáveis no dBASE II e dos comandos referentes às mesmas, tais como STORE, INPUT, ACCEPT e outros.

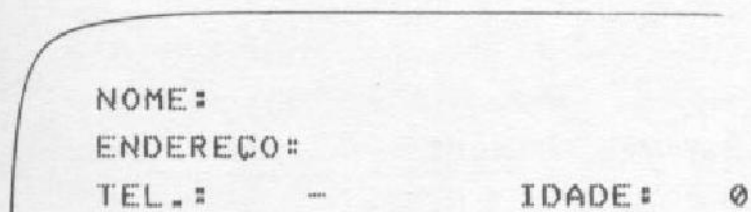
Terminada a digitação, teclar CONTROL e W para encerrar a edição do módulo e gravá-lo no disquete. Em seguida, executá-lo, comandando:

DO PEGDADO <RETURN>

O leitor observará que o programa terá sua execução sustada após gerar a tela de vídeo reproduzida na Figura 5-A.

Notará também que o cursor do sistema estará posicionado no primeiro espaço do "conteúdo" da variável predefinida NOME, exibido pelo comando GET. (Lembremos que a variável NOME recebeu apenas espaço em branco ao ser definida por STORE e que esse é o seu conteúdo por ora.)

Nessa condição, por força do comando READ, o conteúdo da variável NOME pode ser alterado, isto é, o seu espaço em branco pode ser preenchido através de digitação pelos dados que se lhe queira atribuir.

Uma vez digitados os dados e pressionada a tecla RETURN, o cursor saltará automaticamente para a próxima área exibida por GET, com finalidade idêntica e assim sucessivamente.

Depois de processada a última área nessas condições, a execução do programa prossegue automática e normalmente.

```
            DADOS PARA AGENDA


Digite o nome :                          :

Digite o endereço :                          :

     Telefone :    -     : Idade : 0:
```

Figura 5-A

No caso do módulo de programa em exame, sua execução cabal gerará a exibição da parte final da tela de vídeo, com os dados digitados nos campos exibidos por GET e o retorno do ponto de prontidão para comando do dBASE.

A parte restante da tela a ser exibida é reproduzida na Figura 5-B, sem os dados digitados, que dependerão do usuário.

```
NOME:
ENDEREÇO:
TEL.:    -          IDADE:   0
```

Figura 5-B

A fim de que possam ser claramente entendidas as funções de GET e READ, vamos usar o mesmo módulo e fazer nova experiência, eliminando o comando READ. Para tanto, bastará comandar:

MODIFY COMMAND PEGDADO <RETURN>

para que o módulo seja listado no vídeo no modo de edição.

Nessa condição, o cursor deverá ser deslocado com auxílio das teclas de controle CONTROL e C ou X até a linha em que se encontra o comando READ e, em seguida, devem ser pressionadas as teclas CONTROL e T para que ele seja eliminado.

Pressionar em seguida CONTROL e W para que o módulo sejá regravado no disquete.

Finda a regravação, executá-lo novamente:

DO PEGDADO <RETURN>

e observar que, sem o comando READ, sua execução não é interrompida, sendo impossível atribuir dados às variáveis predefinidas.

Aproveitando ainda o mesmo módulo, vamos alterá-lo ligeiramente, objetivando demonstrar também com clareza a função de CLEAR GETS.

Como primeira providência, editar o módulo:

MODIFY COMMAND PEGDADO <RETURN>

Depois de listado no vídeo no modo de edição, efetuar as seguintes alterações nas linhas de programa indicadas:

Na terceira linha:

```
STORE "ASTROGILDO DA SILVA" TO NOME
```

Na quarta linha:

```
STORE "RUA SEM SAíDA, 00" TO END
```

Após a nona linha:

CLEAR GETS

Na linha anterior à linha @ 14,0:

READ

Feitas essas alterações com auxílio das teclas de controle, pressionar CONTROL e W e, em seguida, executar novamente o módulo:

DO PEGDADO <RETURN>

O leitor observará que, na nova condição do módulo em experiência, sua execução será interrompida depois de gerada no vídeo a tela reproduzida na Figura 5-C.

```
            DADOS PARA AGENDA


Digite o nome :ASTROGILDO DA SILVA :

Digite o endereço :RUA SEM SAíDA, 00    :

     Telefone :    -     : Idade : 0:
```

Figura 5-C

O cursor estará posicionado no primeiro espaço do campo telefone exibido por GET e correspondente à variável TEL, significando ser este o primeiro campo destinado a receber dados via digitação. Os precedentes - correspondentes a NOME e END - não foram abrangidos por READ por força do comando CLEAR GETS, inserido no ponto adequado.

Sugerimos que seja feita mais uma experiência com os comandos em questão, a fim de que seja comprovada sua utilização também para alte-

rar conteúdos de variáveis ou de campos de registros que não sejam espaços em branco.

O mesmo módulo PEGDADO poderá ser usado para tal finalidade, bastando eliminar do mesmo o comando CLEAR GETS.

O procedimento a ser adotado para a realização da experiência é basicamente o mesmo utilizado nas experiências anteriores, de maneira que deixamos de descrevê-lo por acreditar que, neste ponto do Curso, o leitor acompanhante já pode realizar a experiência sem dificuldades.

6 – ALTERAÇÃO E ELIMINAÇÃO DE DADOS VIA PROGRAMAS

6.1 – ALTERAÇÃO DE DADOS

Sejam para fins de atualizações ou de correções, as alterações de dados através de programas são feitas no dBASE preferencialmente com o emprego do comando REPLACE, estudado no primeiro volume deste Curso. Mas podem também ser feitas com a utilização de outros comandos, como GET e READ, estudados no Capítulo 5, ou EDIT, BROWSE, CHANGE e UPDATE, estudados também no primeiro volume.

Em matéria de alteração de dados, o emprego do comando adequado depende do tipo de arquivo processado, de sua estrutura e extensão dos campos e de diversos outros fatores, como freqüência das alterações e abrangência das mesmas, nível de operadores etc.

Podemos dizer que cada arquivo de dados requer um processo específico para alterações dos conteúdos de seus campos, em conseqüência do que não é possível estabelecer regras rígidas para sua programação.

Assim, vamos enfocar a questão através da exposição de casos considerados mais incidentes, sugerindo as soluções julgadas mais adequadas para os mesmos.

6.2 - ALTERAÇÕES DE DADOS ATRAVÉS DOS COMANDOS GET E READ

Conforme verificamos no Capítulo 5, os comandos GET e READ usados em conjunto possibilitam efetuar alterações nos conteúdos de campos de registros e de variáveis de memória, embora sejam usados precipuamente para entrada de dados.

Para efetuar alterações nos conteúdos de campos de registros ou de variáveis de memória, basta causar a exibição dos mesmos na tela de vídeo através do comando GET e acionar em seguida o comando READ. Nessa condição, o cursor do sistema é posicionado automaticamente na área ocupada pelo conteúdo exposto, permitindo digitar nesse mesmo local a alteração desejada ou o novo dado. O que permanecer nessa área passa a ser, também de modo automático, o novo conteúdo do campo de registro ou da variável em processamento.

Se, por qualquer razão, for decidido manter sem alterações um dado exibido após a execução de GET e READ, basta pressionar a tecla RETURN no ato. Assim, o dado será confirmado e permanecerá intacto.

6.3 - ALTERAÇÕES DE DADOS ATRAVÉS DOS COMANDOS REPLACE, EDIT, BROWSE E CHANGE

CASO A: Efetuar alterações em diversos ou todos os campos do registro utilizando apenas um ou dois novos dados entrados via teclado.

Nesse caso o uso de REPLACE é o mais indicado, pois, uma vez digitados os novos dados, os quais devem ser atribuídos transitoriamente a uma ou mais variáveis de memória, conforme o caso, as alterações dos conteúdos dos campos do registro são feitas internamente pelo próprio programa. Obviamente, o registro deve ser invocado previamente por comando apropriado. Exemplo:

```
...
ACCEPT "Digite o nome do produto " TO NOME
LOCATE FOR <produto> = NOME
INPUT "Digite o novo preço de custo " TO NPC
REPLACE <custo> WITH NPC
REPLACE <prvenda> WITH NPC * 1.5
REPLACE <pratac> WITH NPC * 1.25
...
```

Observação: As palavras colocadas entre os sinais < e > representam nomes de campos de registros.

CASO B: Efetuar alterações em diversos ou todos os campos do registro com dados específicos para cada um, os quais devem ser digitados um a um.

Em tal caso, tanto o comando REPLACE como EDIT ou CHANGE são indicados, dependendo da conveniência operacional, da estética desejada para a tela de vídeo, do número de campos e de sua extensão etc.

Exemplo com REPLACE:

```
...
ACCEPT "Digite o código do produto " TO MCOD
LOCATE FOR <código> = MCOD
ACCEPT "Digite o novo nome " TO MNOME
ACCEPT "Digite o novo endereço " TO MEND
ACCEPT "Digite o novo tel " TO MTEL
...
REPLACE <nome> WITH MNOME
REPLACE <end> WITH MEND
REPLACE <tel> WITH MTEL
...
```

Exemplo com EDIT:

```
...
INPUT "Digite o nº do registro a alterar " TO N
GOTO N
EDIT #
...
```

ou

```
...
ACCEPT "Digite o código do produto " TO MCOD
LOCATE FOR <cod> = MCOD
EDIT #
...
```

Qualquer dos dois exemplos produz no vídeo a exibição de todos os campos do registro visado, no modo de edição, permitindo que a alteração de cada campo seja feita diretamente via teclado, com auxílio das teclas de controle ativadas pelo comando EDIT.

Exemplo com CHANGE:

```
...
ACCEPT "Digite o nome do cliente " TO MNOME
LOCATE FOR <nome> = MNOME
? "Indique o(s) campo(s) a alterar."
? "Havendo mais de um, separe-os com virgula."
ACCEPT TO CAMPOS
CHANGE &CAMPOS
...
```

Empregado nessa modalidade, CHANGE exibirá no vídeo um campo especificado por vez, com o conteúdo respectivo, e, na linha seguinte, uma das palavras TROCAR ou TO, conforme seja o tipo do campo - de caracteres ou de números -, a fim de que as alterações sejam digitadas em seguida, no modo próprio de seu funcionamento.

CASO C: Efetuar alterações idênticas em um ou mais cam-

pos de todos os registros do arquivo ou apenas daqueles que satisfaçam uma condição determinada.

O comando REPLACE é o mais indicado para esse caso. Exemplo:

```
...
ACCEPT "Novo prazo para faturamento? " TO NPRAZ
ACCEPT "Nova data-limite mensal? " TO NDAT
REPLACE ALL <prazfat> WITH NPRZ [FOR <condição>]
REPLACE ALL <datafim> WITH NDAT [FOR <condição>]
...
```

CASO D: Efetuar alterações específicas, com dados apropriados a cada registro, em um ou mais campos de todos os registros do arquivo ou apenas dos que satisfaçam uma condição determinada.

Nesse caso podem ser utilizados os comandos BROWSE e CHANGE, sendo que apenas CHANGE permite selecionar os registros que atendam a uma exigência determinada.

Exemplo com BROWSE:

```
...
? "Indique o(s) campo(s) a alterar."
? "Havendo mais de um, separe-os com vírgula."
ACCEPT TO MNOME
GOTO TOP
BROWSE FIELD &MNOME
...
```

Com BROWSE empregado nessa condição, os campos indicados de todos os registros do arquivo serão exibidos na tela de vídeo, na modalidade de edição, permitindo que as alterações sejam digitadas individual e diretamente em cada campo.

Exemplo com CHANGE:

```
...
? "Indique o(s) campo(s) a alterar."
? "Havendo mais de um, separe-os com vírgula."
ACCEPT TO MNOME
CHANGE FIELD &MNOME [FOR <condição>]
...
```

Empregado nessa modalidade, CHANGE exibe sucessivamente, um por vez, todos os registros do arquivo ou apenas os que satisfaçam uma determinada condição, com somente os campos especificados e seus respectivos conteúdos, em modo de edição, permitindo que as alterações desejadas sejam digitadas individualmente para cada campo.

6.4 - ALTERAÇÕES DE DADOS ATRAVÉS DO COMANDO UPDATE

Um dos recursos mais interessantes e poderosos oferecidos pelo dBASE para alterações de dados de um arquivo é o que provém do comando UPDATE.

UPDATE possibilita alterar um arquivo de dados a partir de outro arquivo, criado e usado especialmente para essa finalidade, de tal modo que as alterações sejam efetuadas em lotes. Esse processo oferece a conveniência de submeter o arquivo principal de dados a uso menos intenso, minimizando erros e riscos gerais inerentes ao sistema de armazenamento de informações em disquetes.

UPDATE foi descrito e amplamente exemplificado no primeiro volume deste curso. Assim como os demais comandos abordados neste capítulo, deve ser usado sempre que as conveniências que oferecer sejam preponderantes. Todavia, salientamos que os interessantes e poderosos recursos que UPDATE oferece devem ser considerados plenamente pelo programador sempre que as alterações de dados de um arquivo tiverem de ser feitas com acentuada freqüência.

6.5 - ELIMINAÇÃO DE DADOS

A eliminação de dados de arquivo pode ater-se a um ou mais campos de um registro ou abranger todos.

Eliminar dados de campos de registro significa trocar seu conteúdo não nulo por conteúdo nulo, o que equivale a alterar os conteúdos dos campos ou os dados contidos neles.

Assim, para eliminar dados de apenas parte dos campos de um registro, são válidos os comandos e recursos abordados neste capítulo. Por exemplo:

```
...
ACCEPT "Digite o nome do produto " TO NOME
LOCATE FOR <produto> = NOME
REPLACE <campo alfanumérico> WITH ""
REPLACE <campo numérico> WITH 0
...
```

Eliminar os conteúdos de todos os campos de um registro equivale a anular o registro. Nesse caso, deve ser usado o comando DELETE, conforme exemplificação feita a seguir:

```
...
ACCEPT "Qual o item a buscar? " TO IT
LOCATE FOR <item> = IT
...
ACCEPT "Elimina o registro? (S/N) " TO RESP
   IF RESP = "N"
   ERASE
   RETURN
   ENDIF
   IF RESP = "S"
   DELETE
   ENDIF
...
```

Como sabemos, DELETE apenas assinala o registro para eliminação, sendo esta efetivamente feita pelo comando PACK, o qual poderá ser incluído no mesmo módulo, se houver conveniência. Sobre este ponto, convém reler o Capítulo 12 do primeiro volume deste Curso.

Ao terminar a leitura deste capítulo, o seguidor de nosso Curso terá percebido que a alteração de dados de arquivos processada através de programas em dBASE depende inteiramente de comandos e recursos disponíveis no chamado MODO INTERATIVO do sistema. Esse fato patenteia de forma inequívoca o que temos afirmado com freqüência a respeito do MODO PROGRAMÁVEL do dBASE: seu estudo e aplicação só são viáveis após o conhecimento pleno de todos os comandos e recursos disponíveis em seu modo interativo.

O que afirmamos em relação ao tópico tratado neste capítulo aplica-se de forma geral à programação em dBASE, razão por que enfatizamos a necessidade de o iniciante em programação em dBASE aprofundar-se nos estudos dos comandos e recursos do modo interativo do sistema a fim de poder dominar plenamente o modo programável do mesmo e colher de sua aplicação os melhores resultados.

" " "

7 – COMANDOS DE REPETIÇÃO (LAÇO) E DE DESVIO

Uma das principais vantagens do uso de menus em programas em geral é a de facilitar a integração entre o usuário e o programa. Isto quer dizer que, através da exibição do menu do programa, o usuário toma rapidamente conhecimento dos seus recursos e passa a utilizá-los com facilidade, já que, para tanto, basta digitar os símbolos - letras ou números - que os acionam.

Na prática, essa peculiaridade de programas dirigidos por menus permite que qualquer usuário iniciante em computação os opere sem dificuldades, com um mínimo de digitação.

Todavia, para que um menu de opções de programa atenda plenamente a sua finalidade, é preciso que reúna recursos importantes e característicos, entre os quais destacamos:

- dado o comando para sua execução, ele deve manter-se ativo, isto é, "rodando", até que o usuário decida o contrário;
- a escolha de qualquer das opções para execução de comandos ou de módulos por ele supervisionados deve gerar a ação ou processamento correspondentes;
- após a execução de qualquer comando ou módulo de programa por seu intermédio, deve retornar a ele o controle do sistema;
- no caso de o operador digitar erroneamente símbolos não exis-

tentes nas opções oferecidas, sua execução não deve ser interrompida e o usuário deve ser advertido sobre o fato através de uma mensagem exibida no vídeo.

Para que os requisitos descritos sejam atendidos, o programa gerador de menu deve incluir outros comandos além dos que geram sua exibição ou impressão. São eles os comandos de repetição, de desvio condicional e de retorno.

Como tais comandos são utilizados largamente na programação geral em dBASE e, por conseqüência, em programa gerador de menu, serão estudados neste capítulo, a fim de podermos concluir a elaboração do protótipo de programa gerador de menu iniciado no capítulo anterior.

Comando DO WHILE... ENDDO

*** O comando composto DO WHILE... ENDDO possibilita repetir a execução de outro comando, de conjunto de comandos, de programa ou de módulo de programa enquanto for verdadeira uma condição expressa por ele. Sua sintaxe é:

```
DO WHILE <condição>
<comando(s)>
ENDDO
```

Além de a condição expressa no comando ter de ser verdadeira, é imperioso que o(s) comando(s) a ter(em) sua execução repetida seja(m) inserido(s) entre DO WHILE e seu complemento ENDDO, já que é este que indica o ponto de retorno do ciclo de repetição.

O controle de repetição é feito com auxílio de uma variável qualquer, à qual se atribui um valor ou um conteúdo alfanumérico, vinculando-o a DO WHILE como condição verdadeira. Por exemplo:

```
STORE "S" TO REP
DO WHILE REP = "S"
<comando(s)>
ENDDO
```

Nessa situação, isto é, enquanto S for o conteúdo de REP, os comandos incluídos entre DO WHILE e ENDDO serão executados repetidamente e na ordem em que foram incluídos.

Para interromper a ação de DO WHILE... ENDDO é necessário incluir no ciclo repetitivo um comando cuja execução permita alterar opcionalmente o conteúdo de REP, conforme é exemplificado a seguir:

```
STORE "C" TO REP
DO WHILE REP = "C"
(comando(s))
? "Digite C p/continuar ou P p/parar "
ACCEPT TO REP
ENDDO
```

Nessa disposição, após a execução de cada ciclo, o usuário é instado a digitar C ou P, permitindo ou não a repetição de sua execução. Se digitar C, mantém a condição expressa por DO WHILE e a repetição ocorre. Digitando P, altera o conteúdo da variável REP e, conseqüentemente, a condição expressa por DO WHILE, que é desativado, cessando a repetição do circuito.

O controle de repetição pode ser feito também com auxílio de uma variável contadora incluída no circuito repetitivo, conforme exemplo que segue:

```
STORE 0 TO C
DO WHILE C<10
(comando(s))
STORE C+1 TO C
ENDDO
```

Nessa condição, a cada execução dos comandos incluídos entre DO WHILE e ENDDO a variável C é incrementada em uma unidade, até atingir 10, quando então a condição expressa por DO WHILE deixa de ser "verdadeira", cessando a sua ação.

Vários comandos DO WHILE... ENDDO podem ser encadeados.

Em computação, um trecho de programa compreendido entre DO WHILE e ENDDO é denominado "laço" ("loop" em inglês), em razão de permitir que a seqüência normal de execução do programa seja interrompida temporariamente a fim de que os comandos incluídos em seu circuito sejam executados repetitivamente, em conformidade com uma condição expressa ou enunciada no comando.

Todavia, se uma outra condição especial for especificada dentro do laço DO WHILE... ENDDO, seu fluxo de execução pode ser desviado novamente para o seu início, sem execução dos comandos restantes do circuito. Tal desvio é feito pelo comando descrito a seguir:

Comando LOOP

*** LOOP desvia o fluxo de execução de comandos contidos em circuito DO WHILE... ENDDO para o início do mesmo. Sua sintaxe é apenas:

```
LOOP
```

De certa forma o efeito de LOOP é o mesmo de ENDDO, já que a função deste é retornar o fluxo de execução do laço DO WHILE... ENDDO para o seu início, com a finalidade de repetição. A diferença entre ambos reside no fato de que, depois de ativado por determinada condição, LOOP impede que comandos situados entre ele e ENDDO sejam executados.

Exemplo de módulo com DO WHILE... LOOP... ENDDO:

```
DO WHILE <condição>
<comando(s)>
SE QUANTIDADE >100
SKIP
LOOP
FIM
<COMANDO(S)>
ENDDO
```

Nesse exemplo, se a condição QUANTIDADE >100 verificar-se num registro em processamento, os comandos situados após LOOP não serão exe-

cutados em relação a esse registro, que será pulado (SKIP), retornando o fluxo de execução para o início do laço. SE e FIM no exemplo representam o comando IF... ENDIF, que será estudado a seguir.

Comando IF... ELSE... ENDIF

*** IF... ELSE... ENDIF é um comando composto que, condicionalmente, possibilita a execução de comando(s), de programa ou de módulo de programa situados fora da sequência de comandos em que se encontra. Sua sintaxe é:

```
IF <condição>
<comando(s)>
[ELSE]
<comando(s)>
ENDIF
```

A cláusula ELSE é opcional no comando, enquanto que ENDIF é obrigatória porque indica a posição no programa onde termina a ação de IF.

A condição expressa por IF pode ser formulada com caracteres alfanuméricos ou variável alfanumérica e número ou variável numérica.

Um comando IF é executado somente se a condição que expressa for verdadeira. Todavia, deve ser observado que, se incluir a cláusula ELSE e a condição que expressa não for verdadeira, será(ão) executado(s) o(s) comando(s) incluído(s) entre ELSE e ENDIF. Se a condição expressa não for verdadeira e não houver cláusula ELSE no comando, o fluxo de execução salta para a primeira linha de programa após ENDIF.

Como sabemos, normalmente um programa é executado linha por linha de comando, seqüencialmente, isto é, na ordem em que foram digitadas, a menos que existam no mesmo comandos de repetição (DO WHILE... ENDDO, como já vimos) ou comandos como IF... ELSE... ENDIF, com capacidade de desviar o fluxo de execução, desde que verificadas as condições expressas por eles.

Em razão de sua função, IF... ELSE... ENDIF é considerado comando de desvio condicional. O dBASE não possui comandos de desvio in-

condicional, como GOTO e GOSUB do BASIC.

Exemplo de uso de IF... ENDIF:

```
        SET TALK OFF
        ERASE
        @ 7,10 SAY "Digite o preço"
        INPUT TO PR
        IF PR >5000
        @ 10,5 SAY "Muito caro! Melhore a oferta!"
        ENDIF
        SET TALK ON
```

Exemplo de uso com DO WHILE... ENDDO e IF... ELSE... ENDIF

```
        SET TALK OFF
        ERASE
        STORE "S" TO QQ
        STORE 1 TO CONT
RI      DO WHILE QQ = "S"
        @ 5,5 SAY "Digite o preço para o item "
        @ 5,32 SAY CONT USING "99"
        @ 7,0
        INPUT TO PR
C1I        IF PR >5000
           @ 9,10 SAY "Muito caro!"
             ELSE
             @ 9,10 SAY "OK! Preço aceito!"
C1F        ENDIF
        STORE CONT+1 TO CONT
        @ 12,10 SAY "Continua? (S/N)"
        ACCEPT TO RESP
C2I        IF RESP = "S"
           ERASE
           LOOP
             ELSE
             STORE "N" TO QQ
C2F        ENDIF
RF      ENDDO
```

SET TALK ON

Nesse exemplo, temos um laço DO WHILE... ENDDO e dois comandos de desvio condicional com IF... ELSE... ENDIF.

As marcações RI e RF no extremo esquerdo da página não fazem parte das instruções. São usadas apenas como referências. RI e RF assinalam o início e o fim do "laço repetitivo" DO WHILE... ENDDO.

Os sinais C1I e C1F assinalam o início e o fim do primeiro comando de desvio condicional.

Os sinais C2I e C2F fazem o mesmo com relação ao segundo comando de desvio condicional.

Nesse exemplo, que é um pequeno programa, podemos verificar que o ciclo repetitivo constituído por DO WHILE... ENDDO permite atribuir valores a itens cuja quantidade é ilimitada enquanto o conteúdo da variável QQ for a letra maiúscula S, pois esta é a condição que mantém DO WHILE em ação neste caso.

O primeiro comando de desvio condicional tem a finalidade de controlar os valores em processamento, aceitando-os ou rejeitando-os. Se forem incluídos comandos adequados entre IF... ELSE... ENDIF, os valores aceitos poderão eventualmente ser atribuídos a variáveis ou a campos de registro em uso ou processados com finalidades diversas.

Poderão também ser incluídos entre IF... ELSE... ENDIF comandos para execução de outros módulos de programas não incluídos no laço de repetição DO WHILE... ENDDO.

O segundo comando de desvio condicional tem a finalidade de permitir ao usuário interromper o andamento do programa quando desejar. Se digitar N quando o vídeo exibir a pergunta "Continua? (S/N)", fará com que seja válida a alternativa ELSE do comando e o caractere N será atribuído à variável QQ através do comando STORE. Em conseqüência, a condição expressa por DO WHILE deixa de ser verdadeira, cessando a sua ação, imediata e automaticamente.

Neste último exemplo os dois comandos IF... ELSE... ENDIF constituem laços independentes, isto é, não encadeados ou "aninhados".

*** Vários comandos IF... ELSE... ENDIF podem ser encadeados ou "aninhados" num mesmo ciclo repetitivo ou num programa. Exemplo:

```
<comando(s)>
   IF <1a. condição>
   <comando(s)>
   ELSE
     IF <2a. condição>
     <comando(s)>
     ELSE
       IF <3a. condição>
       <comando(s)>
       ELSE
       <comando(s)>
       ENDIF
     ENDIF
   ENDIF
<comando(s)>
```

Nessa condição de uso, cada IF adicional deve ser precedido de um ELSE e para cada IF usado deve haver um ENDIF que lhe corresponda, sem o que o programa não funcionará perfeitamente.

Além de IF... ELSE... ENDIF o dBASE possui outro comando de desvio condicional, descrito a seguir:

Comando DO CASE... OTHERWISE... ENDCASE

*** Esse é outro comando composto do dBASE que possibilita, condicionalmente, a execução de comando(s), de programa ou de módulo de programa situados fora da seqüência de comandos em que se encontra. Sua sintaxe é:

```
DO CASE
  CASE <condição>
```

```
        <comando(s)>
        CASE <condição>
        <comando(s)>
        OTHERWISE
        <comando(s)>
      ENDCASE
```

A cláusula OTHERWISE é opcional no comando, enquanto que ENDCASE é obrigatória porque indica onde termina a ação de DO CASE.

A condição expressa por qualquer CASE pode ser formulada com caracteres alfanuméricos ou variável alfanumérica e número ou variável numérica, como ocorre com IF.

Como vemos, a definição para CASE... OTHERWISE... ENDCASE é idêntica à de IF... ELSE... ENDIF.

Ambos são comandos de desvio condicional, com funções aparentemente idênticas. Todavia, há uma diferença básica entre os dois comandos: DO CASE... OTHERWISE... ENDCASE não pode ser encadeado e permite a inclusão de qualquer quantidade de alternativas ou condições em sua composição, além da criada por OTHERWISE, enquanto que IF... ELSE... ENDIF... pode ser encadeado mas permite a inclusão de uma alternativa ou condição apenas por comando, além da criada por ELSE.

A fim de que sejam percebidas claramente as diferenças entre ambos os comandos, apresentaremos a seguir um módulo de programa elaborado com DO CASE... OTHERWISE... ENDCASE e, em seguida, o mesmo módulo elaborado com IF... ELSE... ENDIF.

Módulo com DO CASE:

```
SET TALK OFF
ERASE
INPUT "Digite um nº entre 1 e 4 " TO A
   DO CASE
     CASE A = 1
     ? "PRIMEIRA ALTERNATIVA"
     CASE A = 2
```

```
      ? "SEGUNDA ALTERNATIVA"
      CASE A = 3
      ? "TERCEIRA ALTERNATIVA"
      OTHERWISE
      ? "ALTERNATIVA OTHERWISE"
    ENDCASE
SET TALK ON
```

Módulo com IF... ELSE... ENDIF:

```
SET TALK OFF
ERASE
INPUT "Digite um nº entre 1 e 4 " TO A
   IF A = 1
   ? "PRIMEIRA ALTERNATIVA"
   ELSE
     IF A = 2
     ? "SEGUNDA ALTERNATIVA"
     ELSE
       IF A = 3
       ? "TERCEIRA ALTERNATIVA"
       ELSE
         ? "ALTERNATIVA ELSE FINAL"
       ENDIF
     ENDIF
   ENDIF
SET TALK ON
```

Observar neste módulo que, além da alternativa ELSE normal do comando principal, cada comando IF adicional é precedido de ELSE, o que equivale a ter um ELSE no lugar de cada ENDIF se cada comando for incluído independentemente em vez de encadeado no módulo.

Parece-nos fácil deduzir que é aconselhável usar DO CASE em vez de IF quando são diversas ou muitas as alternativas a incluir num módulo ou num programa; por outro lado, é necessário usar IF quando houver conveniência ou necessidade de criar alternativas encadeadas.

Outro ponto a ressaltar é que, quando se tratar de oferecer o mí-

nimo de alternativas que qualquer dos dois comandos permite, que são duas, recomenda-se o uso de IF, como pode ser deduzido através do exame dos dois exemplos que seguem:

Exemplo com DO CASE:

```
DO CASE
  CASE A > B
  ? "A é maior do que B"
  OTHERWISE
  ? "B é maior ou igual a A"
ENDCASE
```

Exemplo com IF:

```
IF A > B
? "A é maior do que B"
  ELSE
  ? "B é maior ou igual a A"
ENDIF
```

Finalizando as considerações sobre os comandos de desvio descritos, cumpre-nos observar que é praxe digitá-los nas linhas de programa com encolunamento diferenciado, como nos exemplos reproduzidos, a fim de facilitar sua visibilidade para fins de análise e clareza, especialmente quando houver diversos IFs encadeados no programa.

7.1 - COMPLETANDO O MÓDULO DO MENU COM DUAS VERSÕES

Em seguida vamos completar o menu iniciado no capítulo anterior, já que, com os últimos comandos estudados, temos os conhecimentos necessários para sua elaboração final.

Usaremos as gravações já feitas dos módulos MENU e MENU2.

Com o dBASE ativado e estando alojado no "disk drive" corrente o

disquete que contém os arquivos dos dois módulos citados, comandar:

MODIFY COMMAND MENU (RETURN)

Estando o módulo exibido no vídeo no modo de edição, deslocar com auxílio das teclas de controle o cursor para a quarta linha do programa, onde se encontra o comando ERASE, e pressionar duas vezes simultaneamente as teclas CONTROL e N a fim de inserir nesse local duas linhas em branco, nas quais deverão ser digitados os comandos:

```
STORE 0 TO OP
DO WHILE OP <> 7
```

Em seguida deslocar o cursor para a primeira linha em branco após a última linha de programa existente e digitar:

```
@ 18,0
INPUT TO OP
   IF OP = 1
   DO OPC-1
   ELSE
      IF OP = 2
      DO OPC-2
      ELSE
         IF OP = 3
         DO OPC-3
         ELSE
            IF OP = 4
            DO OPC-4
            ELSE
               IF OP = 5
               DO OPC-5
               ELSE
                  IF OP = 6
                  DO OPC-6
                  ELSE
                     IF OP = 7
                     SET TALK ON
                        ELSE
```

```
                @ 18,12 SAY "Opção inválida!"
                STORE 1 TO TIME
                DO WHILE TIME <50
                STORE TIME+1 TO TIME
                ENDDO
              ENDIF
            ENDIF
          ENDIF
        ENDIF
      ENDIF
    ENDIF
  ENDIF
ENDDO
```

Terminada e conferida a digitação, pressionar as teclas CONTROL e W simultaneamente para encerrar a edição do programa e gravá-lo no disquete automaticamente.

Temos agora pronta uma das versões do programa gerador de menu.

Deve ser observado que as opções 1 a 6 do menu se destinam a executar módulos do programa "CONTROLE DE ESTOQUE" que ainda não foram elaborados e que, portanto, não se encontram gravados no disquete, de modo que, se o módulo for executado e for escolhida uma dessas opções, será gerada uma mensagem de erro no vídeo, sendo também solicitada ao usuário uma resposta Y ou N (sim ou não) quanto a corrigi-lo ou não. Se tal situação ocorrer, isto é, se o módulo for testado antes mesmo de ter os outros módulos prontos, deverá o usuário responder N à referida mensagem, a fim de não abortar a sua execução.

De outro modo, o teste poderá ser feito sem esse problema se, em vez dos números 1 a 7, for digitado qualquer outro número, o que tornará válida a última cláusula ELSE do módulo, gerando no vídeo a mensagem nela incluída, "Opção inválida!", durante alguns segundos, sem interromper a execução do módulo.

Contudo, se o usuário desejar testar efetivamente todas as opções do menu, mesmo antes de ter prontos os módulos referentes às opções de número 1 a 6, poderá fazê-lo do modo que descrevemos a seguir:

- editar o módulo (MODIFY COMMAND MENU <RETURN>);

- digitar em lugar de cada comando DO OPC nº existente após cada IF OP = nº, o seguinte comando: ? "OPC nº";

- encerrar a edição (CONTROL+W);

- executar o módulo (DO MENU <RETURN>);

- efetuar os testes desejados;

- editar novamente o módulo, devolvendo-lhe a forma definitiva.

Como dissemos antes, a execução de um menu só deve ser interrompida por decisão do usuário, recurso esse que foi implementado em nosso menu através dos comandos:

```
STORE 0 TO OP
DO WHILE OP <> 7
ENDDO
```

de sorte que, estando prontos todos os módulos do programa, sua execução deverá ser sustada somente se o usuário escolher a opção número 7, o que poderá ser constatado também através de teste no modo descrito linhas acima.

Os últimos comandos acrescentados no módulo em desenvolvimento já haviam sido descritos, comentados e exemplificados, de sorte que suas funções deverão ser compreendidas sem dificuldades.

O único ponto que nos parece merecer uma ligeira explicação é o que se refere aos comandos da última cláusula ELSE do módulo. Como sabemos, estando o módulo em execução e não sendo escolhida nenhuma das opções nele disponíveis, isto é, sendo digitado um número que não esteja compreendido entre 1 e 7, inclusive, tornar-se-á válida a última cláusula ELSE do mesmo, sendo executados os comandos a ela referentes. Sabemos que o primeiro deles é responsável pela exibição no vídeo da mensagem "Opção inválida!". Os outros quatro comandos constituem um "laço" de repetição DO WHILE... ENDDO cuja função é manter no vídeo a mensagem durante o tempo suficiente para o usuário tomar conhecimento dela. Se forem suprimidos, a mensagem não chegará a ser percebida pelo

usuário, pois o programa entra imediatamente em "looping", isto é, o fluxo de sua execução retorna ao seu início, por força do "laço" principal montado com DO WHILE... ENDDO, esvaziando a tela de vídeo e reimprimindo na mesma o menu com tal rapidez que a mensagem mal poderá ser vista.

Vamos agora completar a segunda versão do menu. Usando o módulo MENU2 gravado no disquete e aplicando a experiência já adquirida, vamos acrescentar-lhe os comandos complementares, de modo que fique conforme a listagem que segue:

```
*** PROGRAMA "CONTROLE DE ESTOQUE" ***
* MÓDULO MENU2 - GERADOR DO MENU DE OPÇÕES
*
SET TALK OFF
ERASE
STORE 0 TO OP
DO WHILE OP <> 7
TEXT



                 CONTROLE DE ESTOQUE
                 -------------------


            1 - ENTRADA DE PRODUTO NOVO
            2 - ENTRADA DE PRODUTO NÃO NOVO
            3 - SAÍDA DE PRODUTO
            4 - DEVOLUÇÃO DE PRODUTO
            5 - PESQUISA DE PRODUTO
            6 - EMISSÃO DE LISTAGEM
            7 - ENCERRA O PROCESSAMENTO

            <<< Digite o número da opção >>>

ENDTEXT
INPUT TO OP
```

```
DO CASE
   CASE OP = 1
   DO OPC-1
   CASE OP = 2
   DO OPC-2
   CASE OP = 3
   DO OPC-3
   CASE OP = 4
   DO OPC-4
   CASE OP = 5
   DO OPC-5
   CASE OP = 6
   DO OPC-6
   CASE OP = 7
   SET TALK ON
     OTHERWISE
     @ 18,12 SAY "Opção inválida!"
     STORE 1 TO TIME
     DO WHILE TIME <50
     STORE TIME+1 TO TIME
     ENDDO
 ENDCASE
ENDDO
```

Depois de terminada, esta segunda versão do módulo de menu poderá ser testada nas mesmas condições descritas para a primeira versão. Todas as observações feitas para aquela são válidas para esta.

As duas versões agora disponíveis darão ao usuário a oportunidade de analisar dois modos diferentes de programação de um mesmo módulo, permitindo-lhe ampliar seus conhecimentos sobre o assunto.

Tendo concluído o módulo MENU, podemos elaborar os demais módulos do programa "CONTROLE DE ESTOQUE". O MENU funcionará como SUPERVISOR ou CONDUTOR (DRIVER) dos mesmos e servirá como guia ou roteiro na sua elaboração. Antes de os iniciarmos, porém, devemos tecer algumas considerações sobre os tópicos MENSAGENS EXIBIDAS NO VÍDEO e RETORNO E CANCELAMENTO EM EXECUÇÃO DE PROGRAMA, o que será feito nos dois breves capítulos seguintes.

8 — MENSAGENS PROGRAMADAS

Mensagens de advertência, de auxílio ou de orientação ao usuário são muito usadas em programação em geral e em dBASE. Podemos dizer que elas são parte integrante da programação. Um programa sem tais mensagens pode exigir muita "adivinhação" ou prática por parte do usuário em certas fases de sua execução, ou então obrigá-lo a fazer freqüentes consultas à sua documentação, se ela existir. Não devemos confundir estas mensagens com as que são próprias do sistema e descritas no item 7.11 do primeiro volume deste Curso.

Quando a execução de um programa é interrompida para entrada de dados via teclado, por exemplo, é importante que o operador saiba imediatamente o que está ocorrendo e o que deve fazer, sem precisar adivinhar. Desse modo, a execução de tarefas para a qual o programa foi elaborado é cumprida com maior rapidez e exige menos esforço do usuário. Portanto, um programa nessas condições é mais eficiente.

Basicamente, existem dois critérios para programar mensagens para exibição na tela de vídeo:

1º - a mensagem deve permanecer exibida até que o usuário pressione uma tecla especificada ou digite um dado solicitado;

2º - a mensagem deve ser exibida na tela de vídeo durante um tempo determinado e desaparecer sem interferência do usuário.

Quando a mensagem se enquadra no primeiro critério, dizemos que é "mensagem com tempo de exibição dependente". A mensagem que se enquadra no segundo critério é designada como "mensagem com tempo de exibição limitado".

8.1 - MENSAGEM COM TEMPO DE EXIBIÇÃO DEPENDENTE

Neste caso, geralmente a mensagem tem a finalidade de solicitar a escolha de uma opção de processamento ou uma entrada de dados e o tempo de sua exibição está relacionado com a espera ocasionada por um dos comandos ACCEPT ou INPUT, usados para criação, introdução e atribuição de variáveis de memória.

No caso de escolha de opções, qualquer dos comandos citados pode ser usado, dependendo apenas de harmonizá-lo no conjunto, isto é, se a escolha de opções for feita através de letra, deverá ser usado ACCEPT; se a escolha for feita por meio de número, deverá ser usado INPUT. Outros comandos como SAY, ? e REMARK podem ser usados para a mesma finalidade, embora ACCEPT e INPUT sejam preferíveis em razão de sua ação ser mais abrangente, já que permitem introduzir mensagens ou comentários ao mesmo tempo que criam e introduzem variáveis relacionadas com os mesmos, para receber atribuições. Exemplos:

```
ACCEPT "Digite a letra da opção" TO REP

INPUT "Digite o número da opção" TO REP
```

Com os comandos SAY, ? e REMARK, comandos equivalentes seriam os exemplificados a seguir:

```
@ 5,10 SAY "Digite a letra da opção"
ACCEPT TO REP

? "Digite o número da opção"
INPUT TO REP

REMARK Digite a opção
```

ACCEPT ou INPUT TO REP

No caso de entrada de dados, a escolha do comando a ser utilizado deve ater-se à classificação do dado. Exemplos:

```
INPUT "Digite a quantidade" TO QT

ACCEPT "Digite o nome" TO NM
```

Devemos lembrar, ainda, que:

- INPUT aceita dados numéricos quando os mesmos são representados unicamente por números e digitados sem aspas. Neste caso, a variável criada por INPUT é classificada automaticamente como numérica e tratada como tal pelo sistema.
- INPUT aceita dados alfanuméricos quando estes são representados por quaisquer caracteres, inclusive números, e digitados entre aspas simples ou duplas. Neste caso a variável criada por INPUT é classificada automaticamente pelo dBASE como alfanumérica.
- ACCEPT aceita apenas dados alfanuméricos, representados por caracteres quaisquer, inclusive os que representam números, devendo os mesmos ser digitados sem aspas. Se forem digitados com aspas, estas serão incluídas como partes integrantes dos dados. A variável criada por ACCEPT é invariavelmente alfanumérica.

O estudo completo das variáveis de memória do dBASE II foi feito no Capítulo 22 do primeiro volume deste Curso.

Há casos em que é conveniente usar ACCEPT em vez de INPUT e processar o valor correspondente através da função VAL. Exemplo:

```
ACCEPT "Digite a quantidade" TO MQT
STORE VAL(MQT) TO QT
```

Tal recurso evita causar interrupção subseqüente na execução do programa quando, em vez de digitar um número ou valor, o usuário pressionar apenas a tecla RETURN, pretendendo significar nenhum valor ou quantidade nula ou zero. Em tal caso, se for usada a instrução INPUT e

for pressionada a tecla RETURN, será gerada no vídeo a mensagem:

Erro de sintaxe, redigitar.

O dBASE dispõe ainda de outro comando que pode ser usado na emissão de mensagens programadas com tempo de exibição dependente. Trata-se de um "comando de espera" específico, introduzido a seguir:

Comando WAIT

*** WAIT interrompe o processamento de um programa até que qualquer tecla seja pressionada. Sua sintaxe é:

WAIT [TO <nome de variável>]

A variável que pode ser criada opcionalmente por WAIT é alfanumérica, isto é, de caracteres, sendo tratada como tal pelo dBASE.

Ao ser executado, WAIT emite no vídeo a seguinte mensagem:

Esperando.

Assim, ao empregar WAIT para emissão de mensagem programada, esta deve preceder a sua execução através de um dos comandos SAY, ? ou REMARK, devendo ser considerado que, invariavelmente, a sua própria mensagem "Esperando." também será exibida na tela de vídeo.

Se a opção [TO <nome de variável>] for incluída no comando, o que for digitado após a sua execução será atribuído à variável criada.

Em idêntica situação, se a tecla RETURN for pressionada, nenhuma atribuição será feita à variável criada.

Em igualdade com os comandos ACCEPT e INPUT, pressionando a tecla ESC após a execução de WAIT, enquanto o sistema estiver "em espera", a execução do programa será cancelada.

8.2 - MENSAGEM COM TEMPO DE EXIBIÇÃO DETERMINADO

Quando a mensagem a ser exibida no vídeo não depender de digitação de qualquer tecla ou de dados para cessar, ela deve ser gerada por intermédio de um dos comandos SAY, ? ou REMARK e mantida no vídeo durante o tempo desejado através do comando composto DO WHILE... ENDDO controlado por uma variável contadora. Exemplo:

```
...
@ 10,10 SAY "Mensagem"
STORE 1 TO CONT
DO WHILE CONT <50
STORE CONT+1 TO CONT
ENDDO
...
```

O tempo de exibição no vídeo é determinado através do comando:

```
DO WHILE CONT <50
```

Aumentando ou diminuindo o valor expresso no comando, a mensagem será mantida no vídeo durante mais ou menos tempo.

Os módulos já elaborados MENU e MENU2 utilizam as duas modalidades de mensagens descritas.

Nos módulos que elaboraremos para completar o programa CONTROLE DE ESTOQUE usaremos tanto quanto possível os recursos propiciados pela inclusão de comentários ou mensagens, de modo a otimizar sua execução e facilidade de operação.

9 — RETORNO E CANCELAMENTO NA EXECUÇÃO DE PROGRAMA

9.1 — RETORNO EM EXECUÇÃO DE PROGRAMA

Na elaboração dos módulos MENU e MENU2 tivemos oportunidade de aprender que os comandos compostos DO WHILE... ENDDO, IF... ELSE... ENDIF e DO CASE... OTHERWISE... ENDCASE permitem a inclusão de outros comandos em seu circuito, inclusive o comando DO, que executa programa ou módulo de programa situado fora da seqüência de execução em que se encontra.

Na maioria dos casos dessa ordem, isto é, quando o comando DO é utilizado para executar um programa ou módulo situado fora do seu bloco, o fluxo de execução é desviado para aquele programa ou módulo externo e, depois de efetuado o processamento previsto, deve retornar ao comando imediatamente seguinte ao que o desviou, isto é, ao primeiro comando situado após o comando DO recém-executado.

As opções normalmente programadas num MENU como o que elaboramos refletem situações como a que acabamos de descrever, isto é, cada comando IF ou cada CASE existente no mesmo, com exceção do de encerramento, inclui em seu circuito um comando DO cuja finalidade é pôr em execução um módulo do programa "CONTROLE DE ESTOQUE" para que efetue o processamento de dados para o qual foi elaborado. Como veremos logo mais, cada módulo desse programa será constituído por um bloco ou con-

junto de comandos separado estruturalmente do MENU mas ligado funcionalmente a ele. Isto significa que a execução de qualquer dos módulos será feita através do MENU, após o que o controle do fluxo de execução do programa deverá retornar a ele.

(Se lembrarmos que a finalidade precípua de um menu de opções de programa é facilitar sua operação, compreenderemos por que o controle do fluxo de execução deve sempre retornar a ele após o processamento de cada módulo. Senão vejamos: qualquer sistema de controle de estoque, por exemplo, deve incluir no mínimo recursos para efetuar ENTRADA, SAÍDA e ATUALIZAÇÃO de dados. Imagine agora o leitor a dificuldade operacional que apresentaria um programa elaborado para essa finalidade sem um MENU para funcionar como SUPERVISOR ou CONDUTOR (DRIVER)!)

Para que o controle do programa retorne ao MENU após a execução de um módulo supervisionado por ele, deve ser incluído no módulo o comando de retorno descrito a seguir:

Comando RETURN

*** RETURN retorna o fluxo de execução de um programa para o comando situado imediatamente após o comando DO que o desviou. Sua sintaxe é o próprio comando:

RETURN

Se um comando RETURN for incluído num módulo ou programa sem um comando DO precedente, existente no mesmo módulo ou noutro, ele interrompe o fluxo de execução onde se encontra, retornando o controle do sistema ao modo interativo do dBASE. Em outras palavras, um RETURN sem DO correspondente provoca o cancelamento na execução de um módulo ou programa.

ATENÇÃO!

Ao ser executado nessa condição, RETURN não fecha arquivos abertos, podendo provocar sua perda parcial ou total, se não forem fechados por um comando subseqüente apropriado antes de o microcomputador ser desligado. Por essa razão recomenda-se usar RETURN somente em cor-

respondência com o comando DO.

Do exposto, deduz-se que RETURN deve ser usado apenas para retornar o fluxo de execução de um programa à sua origem - um menu de opções, por exemplo - após ter sido executado outro módulo ou conjunto de comandos.

Para provocar simplesmente o cancelamento de execução de programa ou de módulo de programa, deve ser usado o comando descrito a seguir.

9.2 - CANCELAMENTO DE EXECUÇÃO DE PROGRAMA

Dependendo da finalidade de um módulo ou programa, pode ser conveniente provocar simplesmente o cancelamento de sua execução após determinado processamento. Tal recurso é propiciado pelo comando descrito a seguir:

Comando CANCEL

*** CANCEL cancela a execução de um módulo ou programa, retornando o controle do sistema ao modo interativo do dBASE. Sua sintaxe é o próprio comando, apenas:

```
CANCEL
```

Ao ser executado, CANCEL também não fecha arquivos abertos, devendo ser tomadas precauções adequadas contra sua perda parcial ou total, gerando o seu fechamento por comando subseqüente apropriado antes de o microcomputador ser desligado.

9.3 - CANCELAMENTO DE EXECUÇÃO ATRAVÉS DE DO WHILE... ENDDO

Outra maneira de provocar o cancelamento de execução de um módulo ou programa é através do comando composto DO WHILE... ENDDO.

Como sabemos, tal comando possibilita repetir a execução de comando(s), de programa ou módulo de programa enquanto uma condição expressa por ele for verdadeira.

Assim, quando a execução do módulo ou programa for controlada pelo comando DO WHILE... ENDDO, é suficiente alterar apenas o fato que condiciona a expressão como verdadeira para que seja imediatamente cancelada sua execução.

Exemplo da utilização desse recurso é encontrado nos módulos MENU e MENU2, na opção número 7, cujo objetivo é provocar o cancelamento da execução do programa. Podemos notar nesse exemplo que é o comando DO WHILE... ENDDO que mantém o módulo em execução e que o que condiciona a expressão enunciada por ele como verdadeira é o fato de o conteúdo da variável OP ser diferente de 7. Assim, ao ser digitado pelo usuário o número 7 como escolha das opções do menu, esse número é atribuído à variável OP automaticamente através do comando INPUT inserido antes da série de comandos IF ou CASE, alterando conseqüentemente aquele fato e provocando o cancelamento da execução do módulo.

10 - ELABORAÇÃO DOS MÓDULOS COMPLEMENTARES DO PROGRAMA "CONTROLE DE ESTOQUE"

Os comandos do dBASE II PLUS estudados até esta parte de nosso Curso já nos permitem concluir a elaboração de um programa-protótipo "CONTROLE DE ESTOQUE", o qual, com ligeiras adaptações para aplicações específicas, poderá ser utilizado na prática sem qualquer dificuldade e com bom desempenho.

Dada sua finalidade didática, todavia, cumpre-nos observar que o programa foi elaborado tendo em vista caracterizá-lo com o máximo possível de clareza e concisão, a fim de tornar fácil o seu entendimento. Isto significa que não foram incluídos no mesmo certos recursos sofisticados que otimizariam o seu desempenho mas que complicariam a sua estruturação geral, dificultando sua compreensão e eliminando a facilidade que deve apresentar para "arranjos" ou adaptações por parte de usuários iniciantes na matéria, a quem concitamos tal prática, que aprimora o aprendizado.

Por outro lado, não seria concebível deixar de descrever e exemplificar em nosso Curso os recursos mais elaborados ou sofisticados do dBASE, com os quais pode contar o programador, mesmo iniciante, para aprimoração dos seus conhecimentos, de sorte que tais recursos serão apresentados e exemplificados depois da elaboração completa do programa "CONTROLE DE ESTOQUE", quando então o seguidor aplicado deste Curso

estará mais apto a assimilá-los. Juntamente com tais recursos serão descritos e estudados os mais recentes comandos implementados no dBASE II, que são os comandos da série PLUS.

Outra questão que convém introduzir neste ponto é a que diz respeito a FLUXOGRAMAS.

10.1 - O USO DE FLUXOGRAMAS EM PROGRAMAÇÃO

Os fluxogramas, também denominados diagramas de blocos, são considerados indispensáveis por alguns programadores para esquematizar graficamente a seqüência de execução de um programa, de modo a constituir a base do mesmo.

Parece-nos não haver dúvidas de que, em muitos casos, o raciocínio para estabelecer o fluxo de operações de forma lógica, fluente e concisa pode ser facilitado pelo uso de fluxogramas.

Todavia, dado o fato de que os fluxogramas lidam quase que exclusivamente com o fluxo de controle dos programas, não permitindo detalhar as operações do processamento propriamente dito dos dados envolvidos e do seu fluxo, parece-nos também que a grande maioria dos programadores os evita atualmente, considerando-os de pouca valia e verdadeiramente "muito chatos".

Por outro lado, depois do advento de linguagens flexíveis como o BASIC, de grande alcance e de extrema inteligibilidade e muita semelhança com o linguajar humano, e de linguagens de programação mais específicas como a do dBASE, a programação de microcomputadores tornou-se muito mais fácil e simples, permitindo que problemas bastante complexos sejam equacionados e programados sem grandes dificuldades e, na maioria dos casos, apenas com auxílio de um roteiro preestabelecido.

Por essas e outras razões, e a critério do programador, os fluxogramas devem ou não constituir elementos básicos ou de simples apoio ao elaborar os seus programas. Mas não podem e não devem ser considerados, por convenção, como elementos básicos da programação de micro-

computadores nem como guias absolutos ou indispensáveis em sua elaboração. Mesmo porque quem não possui muita habilidade para desenhar poderá acabar crendo que nunca aprenderá a elaborar programas para computadores, como se pode deduzir ao examinar a quantidade e diversidade de símbolos existentes para elaboração de fluxogramas.

Concluindo, diremos que a utilização de fluxogramas deve depender do grau de auxílio que podem prestar ao programador em face da complexidade dos programas que deve elaborar.

Com relação aos módulos que comporão nosso programa "CONTROLE DE ESTOQUE", eles serão elaborados sem sustentação em fluxogramas, em razão de estarmos adotando a modalidade TOP DOWN de programação estruturada, comentada no item 3.2 e detalhada no Capítulo 14. Tal modalidade facilita sobremaneira a elaboração de programas, permitindo que qualquer deles seja elaborado praticamente sem o uso de fluxogramas.

Assim sendo, as listagens dos comandos que compõem os módulos do nosso programa "CONTROLE DE ESTOQUE" serão introduzidas sem tais elementos que, todavia, foram elaborados por razões didáticas e se encontram reunidos no item 10.8, para consulta ou estudo.

10.2 - ELABORAÇÃO DO MÓDULO OPC-1 - ENTRADA DE PRODUTO NOVO

Podemos agora elaborar os outros módulos que comporão o programa "CONTROLE DE ESTOQUE".

Como afirmamos antes, o próprio MENU servirá de roteiro para essa tarefa, de sorte que o primeiro dos módulos a elaborar é o OPC-1, correspondente ao item 1 - ENTRADA DE PRODUTO NOVO.

Qualquer módulo criado para essa finalidade em um sistema de controle de estoque é baseado no arquivo de dados (.DBF) em que são armazenados os dados correspondentes aos itens ou produtos controlados por ele. Em verdade, todo o programa do sistema é baseado em tal arquivo, sendo este, via de regra, planificado e criado em primeiro lugar a fim

de que suas características sejam conhecidas e adequadamente consideradas na elaboração do programa gerenciador, de modo que seu conteúdo possa ser devidamente tratado.

Nosso "CONTROLE DE ESTOQUE" será baseado no arquivo CTRLSTOQ.DBF criado no Capítulo 14 do primeiro volume deste Curso, já que o mesmo, tendo sido usado no MODO INTERATIVO para a mesma finalidade, servirá também no MODO PROGRAMÁVEL, possibilitando a formulação de comparações valiosas no aprendizado geral do dBASE.

Estando o dBASE ativado no microcomputador, acionar o editor de programas próprio do mesmo, comandando:

MODIFY COMMAND OPC-1 <RETURN>

e, fazendo uso dos seus recursos, já estudados, digitar a listagem que segue:

```
*** PROGRAMA "CONTROLE DE ESTOQUE" ***
* MÓDULO OPC-1 - ENTRADA DE PRODUTO NOVO
*
SET TALK OFF
ERASE
USE CTRLSTOQ
STORE " " TO OP1
DO WHILE !(OP1) <> "N"
APPEND BLANK
SET CONFIRM ON
@ 1,10 SAY "ENTRADA DE PRODUTO NOVO"
@ 2,10 SAY "======================="
@ 4,12 SAY "Código " GET CODIGO
@ 6,0 SAY "Denominação " GET DENOM
@ 8,0 SAY "Departamento " GET DEP
@ 9,0 SAY "Estoque mínimo " GET QTMIN
@ 10,0 SAY "Quantidade " GET QTAT
@ 11,0 SAY "Preço de custo " GET PRC PICTURE "#####.##"
@ 12,0 SAY "Preço de venda " GET PRV PICTURE "#####.##"
READ
```

```
@ 13,0
REPLACE VALC WITH QTAT * PRC
REPLACE VALV WITH QTAT * PRV
?
ACCEPT "Outra entrada? (<RETURN>=SIM, <N>=NÃO)" TO OP1
ERASE
ENDDO
RELEASE ALL EXCEPT OP
USE
RETURN
```

Terminada e conferida a digitação, pressionar as teclas CONTROL e W simultaneamente, a fim de encerrar a edição do módulo e acionar sua gravação automática no disquete.

Teceremos em seguida alguns comentários sobre esse módulo, a fim de facilitar a sua compreensão:

- O módulo OPC-1 será executado automaticamente sempre que a opção número 1 do MENU for escolhida pelo usuário.

- Ao ser executado, o módulo coloca automaticamente em uso o arquivo CTRLSTOQ.DBF, o qual armazenará a entrada de dados que será feita por seu intermédio, via digitação. Como já comentamos, tal arquivo foi criado no primeiro volume de nosso Curso e deverá estar presente no mesmo disquete em que serão gravados todos os módulos do programa "CONTROLE DE ESTOQUE". O arquivo CTRLSTOQ será a sua base ou sede de dados.

- O laço constituído por DO WHILE !(OP1) <> "N" permite que sejam digitadas entradas de dados para novos registros até que o usuário responda N (= não) em resposta à pergunta "Outra entrada?" exibida ao término da digitação de dados para cada novo produto.

- A forma adotada para a variável OP1 no comando DO WHILE !(OP1) <> "N" tem a função de converter em caractere maiúsculo a resposta atribuída à variável ao fim de cada ciclo, caso seja ela digitada em letra minúscula, o que manteria indevidamente como verda-

deira a condição expressa no comando.

- O comando APPEND BLANK adiciona ao arquivo a cada ciclo do laço um registro em branco, isto é, um novo registro, para que a entrada de dados promovida pelo módulo seja destinada a ele através dos comandos GET e READ.

- Os comandos para entrada de dados para os campos CODIGO, DENOM, DEP, QTMIN e QTAT não incluem a cláusula PICTURE para sua formatação porque seus tamanhos são indicados automaticamente através de sinais : (dois pontos) e correspondem exatamente com a estrutura do registro.

- Os comandos para entrada de dados para os campos PRC e PRV incluem a cláusula PICTURE para indicar ao usuário a possibilidade de digitar os mesmos também com valores decimais.

- Através do comando REPLACE são elaborados automaticamente os dados para os campos VALC e VALV, com base nos dados digitados para os campos QTAT, PRC e PRV.

- O comando RELEASE libera ou apaga quaisquer variáveis de memória usadas durante a execução do módulo, com exceção da variável OP, cujo conteúdo diferente de 7 mantém ativo o módulo MENU.

- Através do comando RETURN o controle do programa retorna ao MENU ao ser encerrado o processamento do módulo OPC-1.

- Este módulo pode ser testado independentemente do módulo MENU, bastando comandar:

```
DO OPC-1   <RETURN>
```

sendo necessário apenas que o arquivo CTRLSTOQ esteja presente no mesmo disquete.

10.3 - ELABORAÇÃO DO MÓDULO

OPC-2 - ENTRADA DE PRODUTO NÃO NOVO

Estando o dBASE ativado no microcomputador, acionar o editor de programas do mesmo, comandando:

MODIFY COMMAND OPC-2 <RETURN>

e, fazendo uso dos seus recursos, digitar a listagem que segue:

```
*** PROGRAMA "CONTROLE DE ESTOQUE" ***
* MÓDULO OPC-2 - ENTRADA DE PRODUTO NÃO NOVO
*
SET TALK OFF
ERASE
USE CTRLSTOQ
STORE " " TO OP2
DO WHILE !(OP2) <> "N"
@ 2,5 SAY "ENTRADA DE PRODUTO NÃO NOVO"
@ 3,5 SAY "==========================="
@ 5,0 SAY "Digite o código ou <RETURN> p/encerrar."
@ 6,0
ACCEPT TO MCOD
   IF MCOD = " "
   USE
   RELEASE ALL EXCEPT OP
   RETURN
   ENDIF
LOCATE FOR CODIGO = !(MCOD)
   IF EOF
   @ 9,8 SAY "Código não encontrado!"
   ?
   ACCEPT "  Pressione <RETURN> para continuar." TO QQ
   RELEASE MCOD
```

```
   ERASE
   LOOP
   ENDIF
ERASE
@ 2,10 SAY "Entrada do produto:"
@ 5,5 SAY "Código " GET CODIGO
@ 7,5 SAY "Produto " GET DENOM
@ 9,5 SAY "Departamento " GET DEP
@ 10,5 SAY "Estoque atual " GET QTAT
@ 11,5 SAY "Estoque mínimo " GET QTMIN
@ 12,5 SAY "Preço de custo " GET PRC
@ 13,5 SAY "Preço de venda " GET PRV
CLEAR GETS
@ 16,5 SAY "Digite os dados de entrada:"
STORE 0 TO MQTAT,MPRC,MPRV
@ 18,0 SAY "Quantidade " GET MQTAT PICTURE "#####"
@ 19,0 SAY "Preço de custo " GET MPRC PICTURE "#####.##"
@ 20,0 SAY "Preço de venda " GET MPRV PICTURE "#####.##"
READ
REPLACE QTAT WITH QTAT+MQTAT
REPLACE PRC WITH MPRC
REPLACE PRV WITH MPRV
REPLACE VALC WITH QTAT * PRC
REPLACE VALV WITH QTAT * PRV
ACCEPT "Outra entrada? (<RETURN>=SIM, <N>=NÃO)" TO OP2
ERASE
ENDDO
RELEASE ALL EXCEPT OP
USE
RETURN
```

Terminada e conferida a digitação, pressionar as teclas CONTROL e W simultaneamente para encerrar a edição do módulo e acionar automaticamente a sua gravação no disquete.

COMENTÁRIOS SOBRE O MÓDULO OPC-2:

- O módulo OPC-2 será executado automaticamente sempre que a op-

ção número 2 do MENU for escolhida pelo usuário.

- Ao ser acionado, o módulo coloca automaticamente em uso o arquivo CTRLSTOQ.DBF e solicita ao usuário que digite o código do produto cujo registro deverá ser atualizado.

- O laço constituído por DO WHILE !(OP2) <> "N" mantém o módulo em execução até que o usuário digite N (ou n) em resposta à pergunta "Outra entrada?" exibida no vídeo ao fim da digitação de dados para cada registro.

- O registro ou "ficha eletrônica" do item que deverá receber os novos dados é "buscado" no arquivo CTRLSTOQ pelo comando LOCATE quando o seu código é digitado pelo usuário após ser exibida no vídeo a mensagem "Digite o código...". Se tal registro não existir no arquivo, a ocorrência será comunicada também através de mensagem exibida no vídeo. Nesse caso, a entrada de dados deverá ser feita como produto novo, através da opção número 1 do MENU.

- Da mesma forma que ocorre com a variável OP, a variável MCOD é usada aqui com a função ! (função "letras maiúsculas", estudada no primeiro volume deste Curso), a fim de que a busca feita pelo comando LOCATE não redunde infrutífera quando o código for representado por letras maiúsculas, como costuma ser, e o código para pesquisa for digitado com letras minúsculas. Os códigos do arquivo CTRLSTOQ estão representados por caracteres numéricos (não números dígitos!), já que os campos que os contêm são alfanuméricos ou de caracteres, de forma que tal problema não ocorre nesse caso, pois esses caracteres só têm um formato. A função será válida se forem usadas letras em vez de caracteres numéricos.

- Os dados dos campos do registro em processamento, com exceção dos campos VALC e VALV, são exibidos no vídeo através do comando GET, para controle do usuário. Neste caso também não é utilizada a cláusula PICTURE porque, além de os dados serem exibidos exatamente como estão armazenados nos respectivos campos, o uso de GET é feito para saída (exibição) de dados e não para entrada, quando então aquela cláusula tem grande serventia.

- Os novos dados solicitados através do comando @ x,y SAY e GET e digitados através do comando READ são processados e locados nos respectivos campos através do comando REPLACE. Os campos VALC e VALV são também atualizados, automaticamente, em função dos novos dados digitados.

- Os comandos finais deste módulo são idênticos aos do módulo anterior e já foram comentados.

- Igualmente ao módulo OPC-1, este também poderá ser testado independentemente do MENU, bastando comandar:

```
DO OPC-2   <RETURN>
```

sendo necessária a presença do arquivo CTRLSTOQ.DBF no disquete.

10.4 - ELABORAÇÃO DO MÓDULO OPC-3 - SAÍDA DE PRODUTO

Com o dBASE ativo no microcomputador, comandar:

```
MODIFY COMMAND OPC-3   <RETURN>
```

a fim de acionar o seu editor de programas. Em seguida, digitar a listagem que segue:

```
*** PROGRAMA "CONTROLE DE ESTOQUE" ***
* MÓDULO OPC-3 - SAÍDA DE PRODUTO
*
SET TALK OFF
ERASE
USE CTRLSTOQ
STORE " " TO OP3
DO WHILE !(OP3) <> "N"
@ 2,11 SAY "SAÍDA DE PRODUTO"
@ 3,11 SAY "================"
```

```
@ 5,0 SAY "Digite o código ou <RETURN> p/encerrar."
@ 6,0
ACCEPT TO MCOD
   IF MCOD = " "
   USE
   RELEASE ALL EXCEPT OP
   RETURN
   ENDIF
LOCATE FOR CODIGO = !(MCOD)
   IF EOF
   @ 9,8 SAY "Código não encontrado!"
   ?
   ACCEPT "  Pressione <RETURN> para continuar." TO QQ
   RELEASE MCOD
   ERASE
   LOOP
   ENDIF
ERASE
@ 2,11 SAY "Saída do produto:"
@ 5,5 SAY "Código " GET CODIGO
@ 7,5 SAY "Produto " GET DENOM
@ 9,5 SAY "Departamento " GET DEP
@ 10,5 SAY "Estoque atual" GET QTAT
@ 11,5 say "Estoque mínimo " GET QTMIN
@ 12,5 SAY "Preço de custo " GET PRC
@ 13,5 SAY "Preço de venda " GET PRV
CLEAR GETS
?
?
INPUT "      Digite a quantidade saída " TO MQTAT

REPLACE QTAT WITH QTAT-MQTAT
REPLACE VALC WITH QTAT * PRC
REPLACE VALV WITH QTAT * PRV
?
ACCEPT "Outra saída? (<RETURN>=SIM, <N>=NÃO)" TO OP3
ERASE
ENDDO
RELEASE ALL EXCEPT OP
```

```
USE
RETURN
```

Terminada e conferida a digitação, pressionar as teclas CONTROL e W simultaneamente para encerrar a edição do módulo e causar a sua gravação automática no disquete.

COMENTÁRIOS SOBRE O MÓDULO OPC-3:

De modo geral, aplicam-se a este módulo os comentários já tecidos sobre o módulo OPC-2. A diferença básica entre este e o anterior é que o OPC-3 solicita e processa dados para a modalidade de saída de produto, enquanto que o OPC-2 o faz para a modalidade de entrada, requerendo a mais os dados de preço de custo e de venda.

10.5 - ELABORAÇÃO DO MÓDULO OPC-4 - DEVOLUÇÃO DE PRODUTO

Estando o dBASE ativo no microcomputador, comandar:

```
MODIFY COMMAND OPC-4   <RETURN>
```

Em seguida, fazendo uso dos recursos do editor de programas já ativado, digitar a listagem que segue:

```
*** PROGRAMA "CONTROLE DE ESTOQUE" ***
* MÓDULO OPC-4 - DEVOLUÇÃO DE PRODUTO
*
SET TALK OFF
ERASE
USE CTRLSTOQ
STORE " " TO OP4
STORE "     " TO MED
DO WHILE !(OP4) <> "N"
@ 2,9 SAY "DEVOLUÇÃO DE PRODUTO"
```

```
@ 3,9 SAY "======================"
@ 5,0 SAY "Digite o código ou <RETURN> p/encerrar."
?
ACCEPT TO MCOD
   IF MCOD = " "
   USE
   RELEASE ALL EXCEPT OP
   RETURN
   ENDIF
LOCATE FOR CODIGO = !(MCOD)
   IF EOF
   @ 9,8 SAY "Código não encontrado!"
   ?
   ACCEPT "  Pressione <RETURN> para continuar." TO QQ
   RELEASE MCOD
   ERASE
   LOOP
   ENDIF
ERASE
@ 2,9 SAY "Produto devolvido:"
@ 5,5 SAY "Código " GET CODIGO
@ 7,5 SAY "Produto " GET DENOM
@ 9,5 SAY "Departamento " GET DEP
@ 10,5 SAY "Estoque atual " GET QTAT
@ 11,5 SAY "Estoque mínimo " GET QTMIN
@ 12,5 SAY "Preço de custo " GET PRC
@ 13,5 SAY "Preço de venda " GET PRV
CLEAR GETS
?
?
INPUT "      Quantidade devolvida " TO MQTAT
REPLACE QTAT WITH QTAT + MQTAT
REPLACE VALC WITH QTAT * PRC
REPLACE VALV WITH QTAT * PRV
?
?
ACCEPT "Mais devolução? (<RETURN>=SIM, <N>=NÃO)" TO OP4
ERASE
ENDDO
```

```
RELEASE ALL EXCEPT OP
USE
RETURN
```

Terminada e conferida a digitação, pressionar simultaneamente as teclas CONTROL e W para encerrar a edição do módulo e causar a gravação automática do mesmo no disquete.

COMENTÁRIOS SOBRE O MÓDULO OPC-4:

Aplicam-se a este módulo os comentários feitos sobre os módulos precedentes.

Este módulo é semelhante ao módulo OPC-2, com a diferença de que requer a digitação apenas da quantidade devolvida do produto, não sendo necessários os dados referentes a preços de custo e de venda, exigidos no módulo OPC-2 porque geralmente tais dados são alterados por ocasião de entradas de produtos.

10.6 - ELABORAÇÃO DO MÓDULO OPC-5 - PESQUISA DE PRODUTO

Com o dBASE ativado no microcomputador, acionar o editor de programas próprio, comandando:

```
MODIFY COMMAND OPC-5   <RETURN>
```

Digitar em seguida a listagem que segue:

```
*** PROGRAMA "CONTROLE DE ESTOQUE" ***
* MóDULO OPC-5 - PESQUISA DE PRODUTO
*
SET TALK OFF
ERASE
USE CTRLSTOQ
```

```
STORE " " TO OP5
DO WHILE !(OP5) <> "N"
@ 2,9 SAY "PESQUISA DE PRODUTO"
@ 3,9 SAY "===================".
@ 5,0 SAY "Digite o código ou <RETURN> p/encerrar."
@ 6,0
ACCEPT TO MCOD
   IF MCOD = " "
   USE
   RELEASE ALL EXCEPT OP
   RETURN
   ENDIF
LOCATE FOR CODIGO = !(MCOD)
   IF EOF
   @ 9,8 SAY "Código não encontrado!"
   ?
   ACCEPT "  Pressione <RETURN> para continuar." TO QQ
   RELEASE MCOD
   ERASE
   LOOP
   ENDIF
ERASE
@ 2,9 SAY "Produto pesquisado:"
@ 5,5 SAY "Código " GET CODIGO
@ 7,5 SAY "Produto " GET DENOM
@ 9,5 SAY "Departamento " GET DEP
@ 10,5 SAY "Estoque atual " GET QTAT
@ 11,5 SAY "Estoque mínimo " GET QTMIN
@ 12,5 SAY "Preço de custo " GET PRC
@ 13,5 SAY "Preço de venda " GET PRV
CLEAR GETS
@ 15,0 SAY "Valor de custo do estoque:"
@ 15,28 SAY VALC USING "#,###,###.##"
@ 16,0 SAY "Valor de venda do estoque:"
@ 16,28 SAY VALV USING "#,###,###.##"
?
?
ACCEPT "Outra pesquisa? (<RETURN>=SIM, <N>=NÃO)" TO OP5
ERASE
```

```
ENDDO
RELEASE ALL EXCEPT OP
USE
RETURN
```

Terminada e conferida a digitação, pressionar as teclas CONTROL e W simultaneamente para encerrar a edição do módulo e causar a sua gravação automaticamente no disquete.

COMENTÁRIOS SOBRE O MÓDULO OPC-5:

De forma geral, os comentários tecidos sobre os módulos precedentes são válidos para este.

Há apenas duas pequenas particularidades com relação a este módulo: a primeira é que não requer digitação de dados para processamento; solicita apenas o código do produto para pesquisa e busca do registro que o contém. A segunda particularidade é que, além dos dados normalmente exibidos no vídeo através dos outros módulos, este produz também a exibição dos campos VALC e VALV que, como sabemos, representam os valores totais de custo e de venda de cada produto, dados esses desejáveis no caso de pesquisa dos itens.

10.7 - ELABORAÇÃO DO MÓDULO OPC-6 - EMISSÃO DE LISTAGEM

Finalmente, chegamos ao último módulo de nosso programa-protótipo "CONTROLE DE ESTOQUE".

Este módulo é composto de duas partes: o módulo principal, encarregado do processamento èspecífico para o qual foi criado, e um submódulo, cuja única função é promover a impressão do cabeçalho da listagem no topo ou no início de cada página.

Quando se deseja imprimir o cabeçalho de um relatório ou listagem apenas na sua primeira página, o submódulo pode ser incluído no módu-

lo do qual deriva.

Quando, ao contrário, pretende-se imprimir o cabeçalho em todas as páginas, impõe-se como solução mais prática criar um submódulo e acioná-lo através do módulo principal com o comando DO <nome do submódulo>. Esse foi o recurso adotado em nosso programa.

Vamos agora digitar as duas partes do módulo OPC-6. Depois de digitados, teceremos outros comentários.

Estando o dBASE ativado no microcomputador, comandar:

```
MODIFY COMMAND OPC-6   <RETURN>
```

e digitar em seguida a listagem que segue:

```
*** PROGRAMA "CONTROLE DE ESTOQUE" ***
* MóDULO OPC-6 - EMISSÃO DE LISTAGEM
*
SET TALK OFF
ERASE
STORE 1 TO PG
STORE 50 TO MÁX
SET FORMAT TO PRINT
DO OPC-6SUB
USE CTRLSTOQ
DO WHILE .NOT. EOF
   IF *
   SKIP
   LOOP
   ENDIF
@ LIN,0 SAY CODIGO
@ LIN,5 SAY DENOM
@ LIN,26 SAY DEP
@ LIN,30 SAY QTMIN USING "####"
@ LIN,35 SAY QTAT USING "#####"
@ LIN,41 SAY PRC USING "#####.##"
@ LIN,50 SAY PRV USING "#####.##"
```

```
@ LIN,59 SAY VALC USING "#######.##"
@ LIN,70 SAY VALV USING "#######.##"
SKIP
STORE LIN + 1 TO LIN
   IF LIN > MAX
   STORE PG + 1 TO PG
   SET EJECT ON
   EJECT
   DO OPC-6SUB
   SET EJECT OFF
   ENDIF
ENDDO
STORE LIN+2 TO LIN
@ LIN+1,9 SAY "***** TOTAIS PARCIAIS *****"
@ LIN+3,5 SAY "DEP"
@ LIN+3,10 SAY "VALOR DE CUSTO"
@ LIN+3,26 SAY "VALOR DE VENDA"
@ LIN+4,5 SAY "====================================="
SUM VALC FOR DEP = "CAM" TO CUSCAM
SUM VALV FOR DEP = "CAM" TO VENCAM
SUM VALC FOR DEP = "BAZ" TO CUSBAZ
SUM VALV FOR DEP = "BAZ" TO VENBAZ
SUM VALC FOR DEP = "SAP" TO CUSSAP
SUM VALV FOR DEP = "SAP" TO VENSAP
STORE CUSCAM+CUSBAZ+CUSSAP TO TOT1
STORE VENCAM+VENBAZ+VENSAP TO TOT2
@ LIN+6,5 SAY "CAM"
@ LIN+6,12 SAY CUSCAM USING "#,###,###.##"
@ LIN+6,28 SAY VENCAM USING "#,###,###.##"
@ LIN+7,5 SAY "BAZ"
@ LIN+7,12 SAY CUSBAZ USING "#,###,###.##"
@ LIN+7,28 SAY VENBAZ USING "#,###,###.##"
@ LIN+8,5 SAY "SAP"
@ LIN+8,12 SAY CUSSAP USING "#,###,###.##"
@ LIN+8,28 SAY VENSAP USING "#,###,###.##"
@ LIN+9,5 SAY "====================================="
@ LIN+11,9 SAY "****** TOTAIS GERAIS ******"
@ LIN+13,5 SAY "Cz$"
@ LIN+13,12 SAY TOT1 USING "#,###,###.##"
```

```
@ LIN+13,28 SAY TOT2 USING "#,###,###.##"
@ LIN+14,5 SAY "========================================="
RELEASE ALL EXCEPT OP
SET FORMAT TO SCREEN
RETURN
```

Terminada e conferida a digitação, pressionar simultaneamente as teclas CONTROL e W para encerrar a edição da parte principal do módulo OPC-6 e acionar sua gravação automática no disquete.

Completada a gravação, comandar:

MODIFY COMMAND OPC-6SUB <RETURN>

e digitar a listagem do submódulo, que segue:

```
*** PROGRAMA "CONTROLE DE ESTOQUE" ***
* MóDULO OPC-6SUB - CABEÇALHO DA LISTAGEM
*
STORE 0 TO LIN
@ 1.0 SAY "* NOME DA EMPRESA *"
@ 1.25 SAY "LISTAGEM DO ESTOQUE"
@ 1,50 SAY "EMISSÃO"
@ 1,58 SAY DATE ()
@ 1,71 SAY "PÁGINA"
@ 1,78 SAY PG USING "##"
@ 2,0 SAY "#======================================="
@ 2,40 SAY "=======================================#"
@ 4.0 SAY "COD"
@ 4,5 SAY "PRODUTO"
@ 4.26 SAY "DEP"
@ 4,31 SAY "MIN"
@ 4,35 SAY "ESTOQ"
@ 4,41 SAY "PRCCUSTO"
@ 4.50 SAY "PRCVENDA"
@ 4,59 SAY "VALORCUSTO"
```

```
@ 4,70 SAY "VALORVENDA"
STORE 6 TO LIN
RETURN
```

Terminada e conferida a digitação, pressionar as teclas CONTROL e W simultaneamente para encerrar a edição do submódulo.

COMENTARIOS SOBRE O MóDULO OPC-6:

- A função do módulo OPC-6 é direcionar para uma impressora acoplada ao microcomputador dados armazenados no arquivo CTRLSTOQ.

- A impressão é comandada pelo comando @ x,y SAY e o direcionamento para a impressora é feito pelo comando SET FORMAT TO PRINT.

- A instrução SET PRINT ON não foi usada porque, no caso do módulo em questão, não há necessidade de enviar instruções especiais à impressora. Se a largura da impressão da listagem excedesse o limite do formulário padrão para impressora de 80 colunas, equivalente a 8 polegadas, seria necessário utilizar a modalidade de impressão condensada, a qual permite imprimir no mesmo formulário até 132 ou 160 caracteres por linha, incluídos os nulos ou "brancos", no caso da impressora MONICA EI 6011. Em tal caso, as instruções indicadas a seguir deveriam ser incluídas no módulo, precedendo a instrução SET FORMAT TO PRINT:

```
SET PRINT ON
? CHR (15)
```

para imprimir em até 132 colunas por linha, ou:

```
SET PRINT ON
? CHR (30) + CHR (53)
```

para imprimir em até 160 colunas por linha.

- Os dados são impressos na forma de listagem ou relatório e os valores globais de custo e de venda de cada produto são totalizados e sumarizados ao final por departamento atualmente existente no campo DEP do arquivo.

- Através das variáveis de memória MAX e LIN é controlada a paginação da listagem, enquanto que a variável PG é usada para numerar suas páginas.

- O comando DO OPC-6SUB inserido nas primeiras linhas do módulo principal aciona a impressão do cabeçalho no início da listagem. Mais abaixo, o mesmo comando aciona a impressão do cabeçalho sempre que a listagem se estenda para uma nova página, o que ocorre após o avanço de cada 50 linhas pela impressora, em conformidade com o valor atribuído à variável MAX.

- Tanto no módulo principal como no submódulo é usada a variável LIN para controlar o avanço do papel na impressora, de modo a ser obtida a paginação desejada.

- O comando USE CTRLSTOQ abre o arquivo para uso dos dados armazenados enquanto que o comando SKIP controla a utilização seqüencial dos seus registros.

- O laço constituído pelo comando DO WHILE .NOT. EOF é responsável pela leitura dos registros até que o fim do arquivo seja atingido, quando então cessa automaticamente a sua atuação.

- O comando composto

```
IF *
SKIP
LOOP
ENDIF
```

faz com que registros marcados para deleção sejam "pulados", a fim de não figurarem no relatório.

- As instruções @ x,y SAY determinam os títulos e dados a serem impressos na listagem.

- A cláusula USING determina o formato de impressão dos dados.

- A totalização e sumarização ao final da listagem são feitas por

intermédio dos comandos SUM e STORE.

- Ao terminar a impressão da listagem, o comando SET FORMAT TO SCREEN redireciona a saída do sistema para a tela de vídeo e o controle do programa é devolvido ao MENU pelo comando RETURN.

Terminada a digitação do módulo OPC-6, está pronto todo o programa-protótipo "CONTROLE DE ESTOQUE". Se sua digitação foi feita em conformidade com as listagens apresentadas, ele poderá ser usado sem problemas em seguida. Deve ser lembrado, todavia, que o programa foi elaborado com base no arquivo CTRLSTOQ, criado no primeiro volume deste Curso e que, conseqüentemente, seu uso deve girar em torno daquele arquivo. Poderá ser usado com arquivo diferente do citado, desde que seja convenientemente adaptado.

Para pôr em execução o programa, deve ser dado o comando

```
DO MENU   <RETURN>
```

depois de ativado o dBASE de acordo com as normas já descritas. Todos os módulos que compõem o programa e mais o arquivo CTRLSTOQ.DBF devem estar reunidos no mesmo disquete, juntamente com os programas de sistema do dBASE II. Os programas que constituem propriamente o sistema dBASE II PLUS PARA MSX são os seguintes:

```
DBASE.COM
DBASEOVR.COM.
```

No caso de haver dois "disk drives" ligados ao microcomputador, a localização dos programas deverá ser feita de modo a dispor do sistema mais convenientemente.

A fim de que sejam suficientemente conhecidos todos os recursos oferecidos pelo programa recém-elaborado, é aconselhável que o mesmo seja exaustivamente testado com base no seu menu de opções e no tipo de arquivo que o mesmo controla.

Depois de testado e conhecido o programa em referência, o seguidor de nosso Curso constatará que o mesmo reúne praticamente todos os

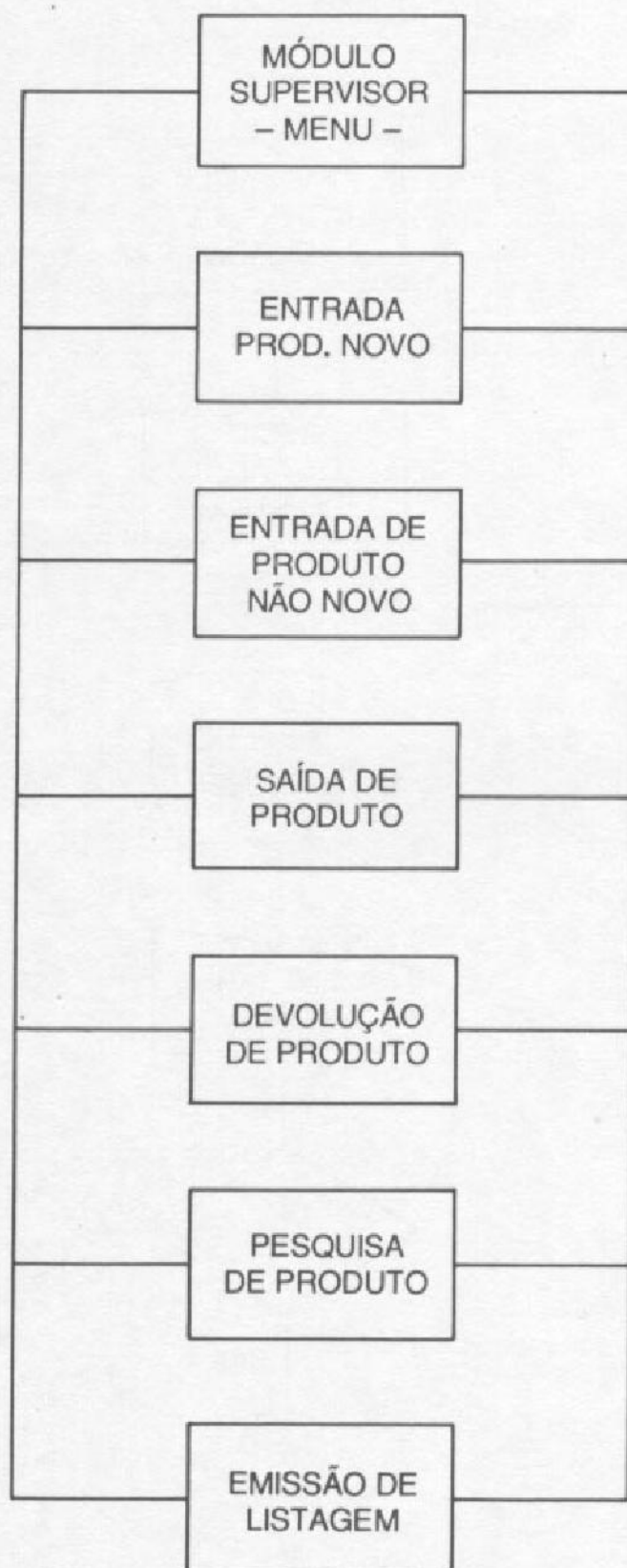
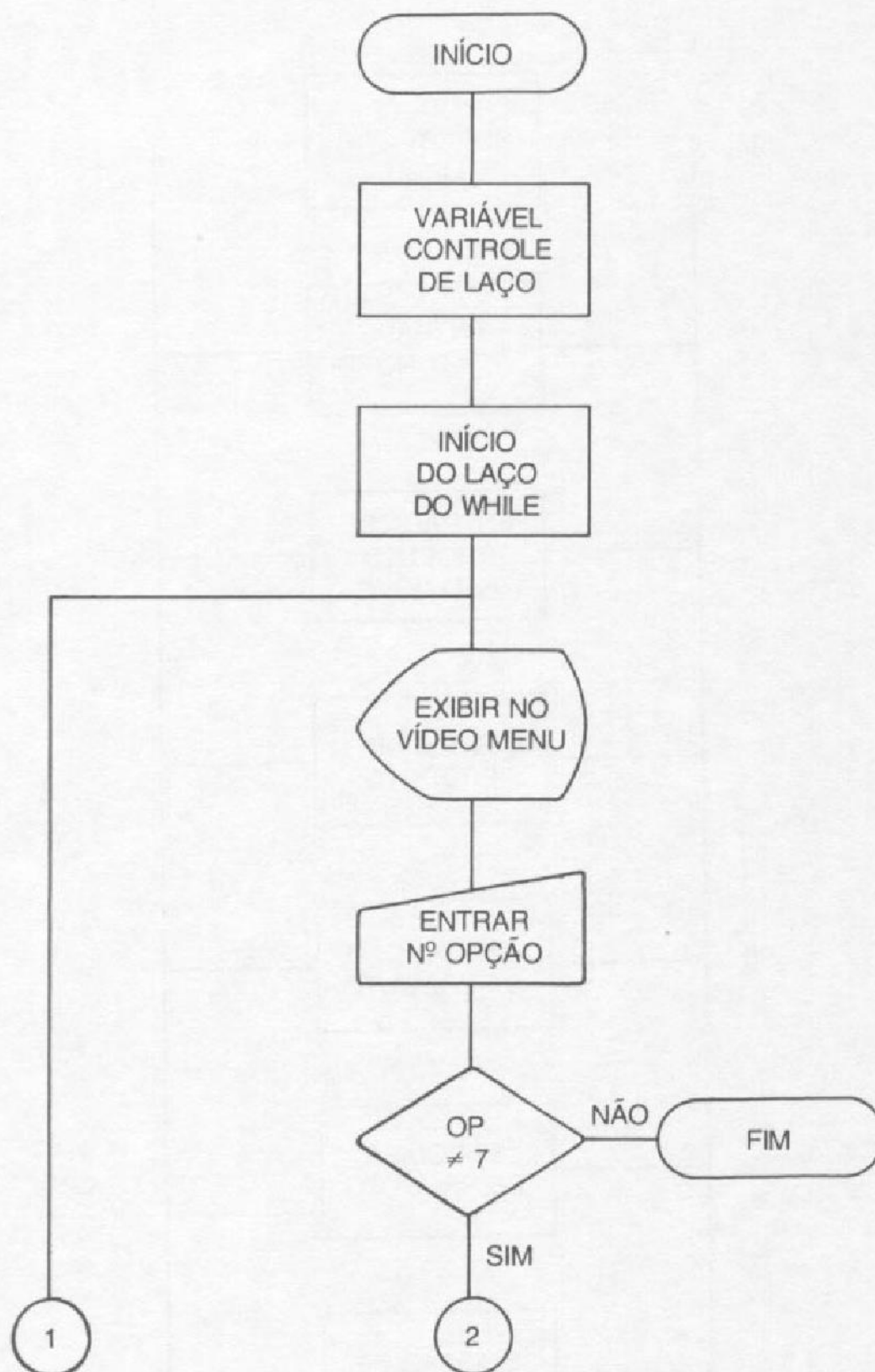
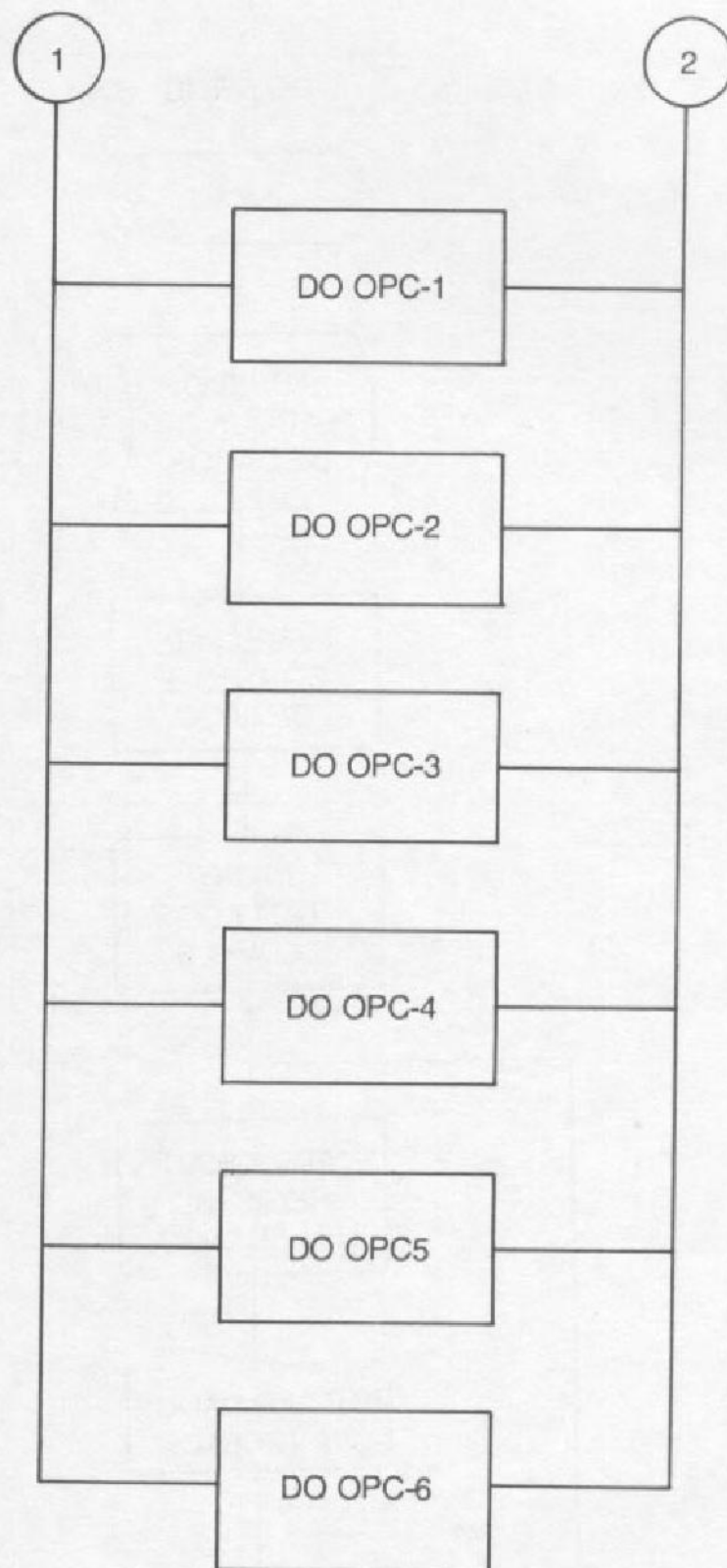
recursos básicos exigidos para a finalidade, podendo servir de modelo ou guia para a elaboração de programas específicos para a área de controle de estoques.

Todavia, e como esclarecemos antes, o programa por nós desenvolvido não inclui certos recursos sofisticados que poderiam dificultar o seu entendimento por iniciantes na matéria. Tais recursos serão apresentados a partir do capítulo seguinte, com a introdução dos comandos da série PLUS e alguns outros, completando o elenco disponível no sistema dBASE II PLUS PARA MSX.

10.8 – FLUXOGRAMAS DO PROGRAMA "CONTROLE DE ESTOQUE"

O nível de detalhamento dos fluxogramas apresentados a seguir é considerado satisfatório. Poderá ser limitadamente elevado ou rebaixado, dependendo da conveniência do programador.

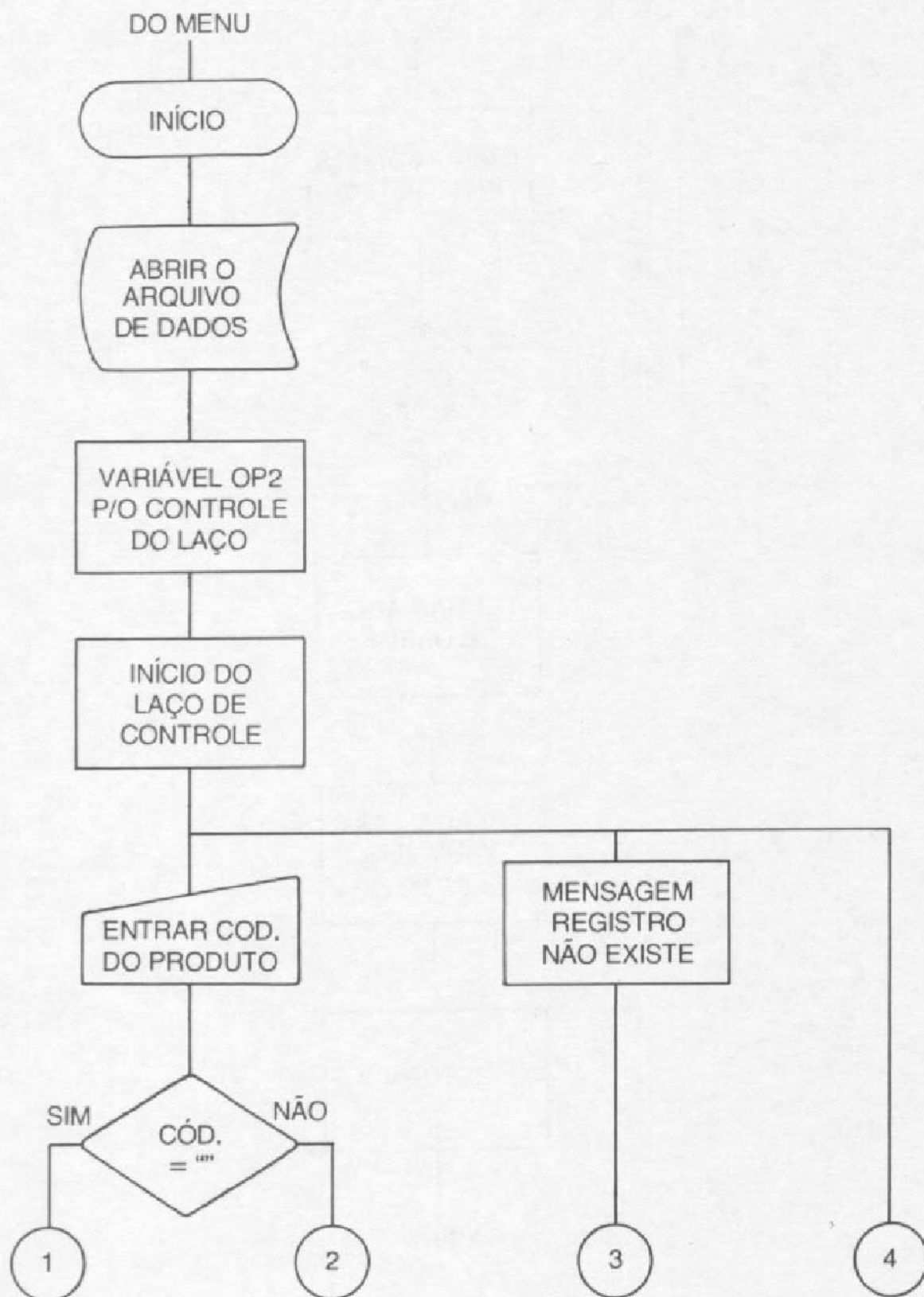
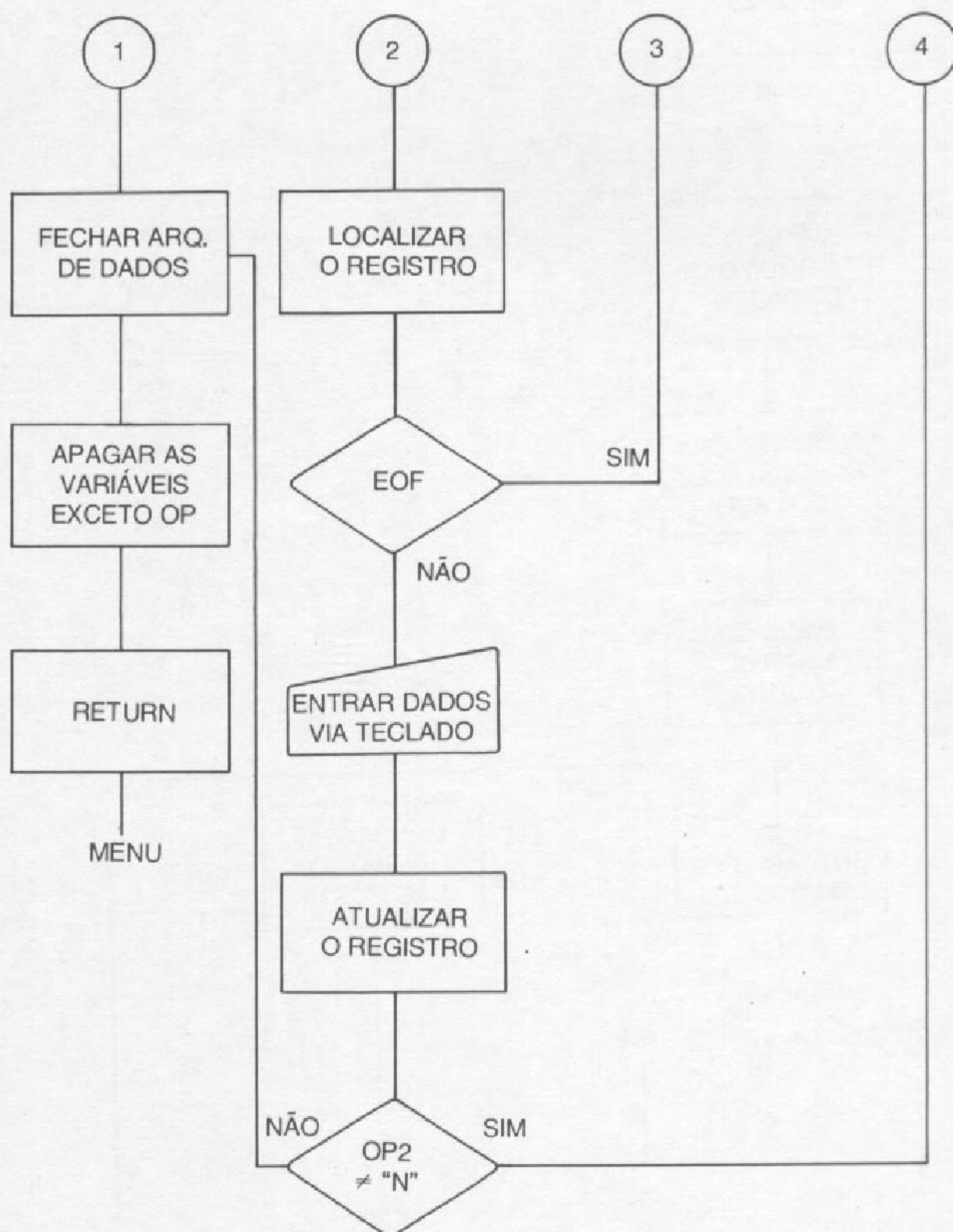
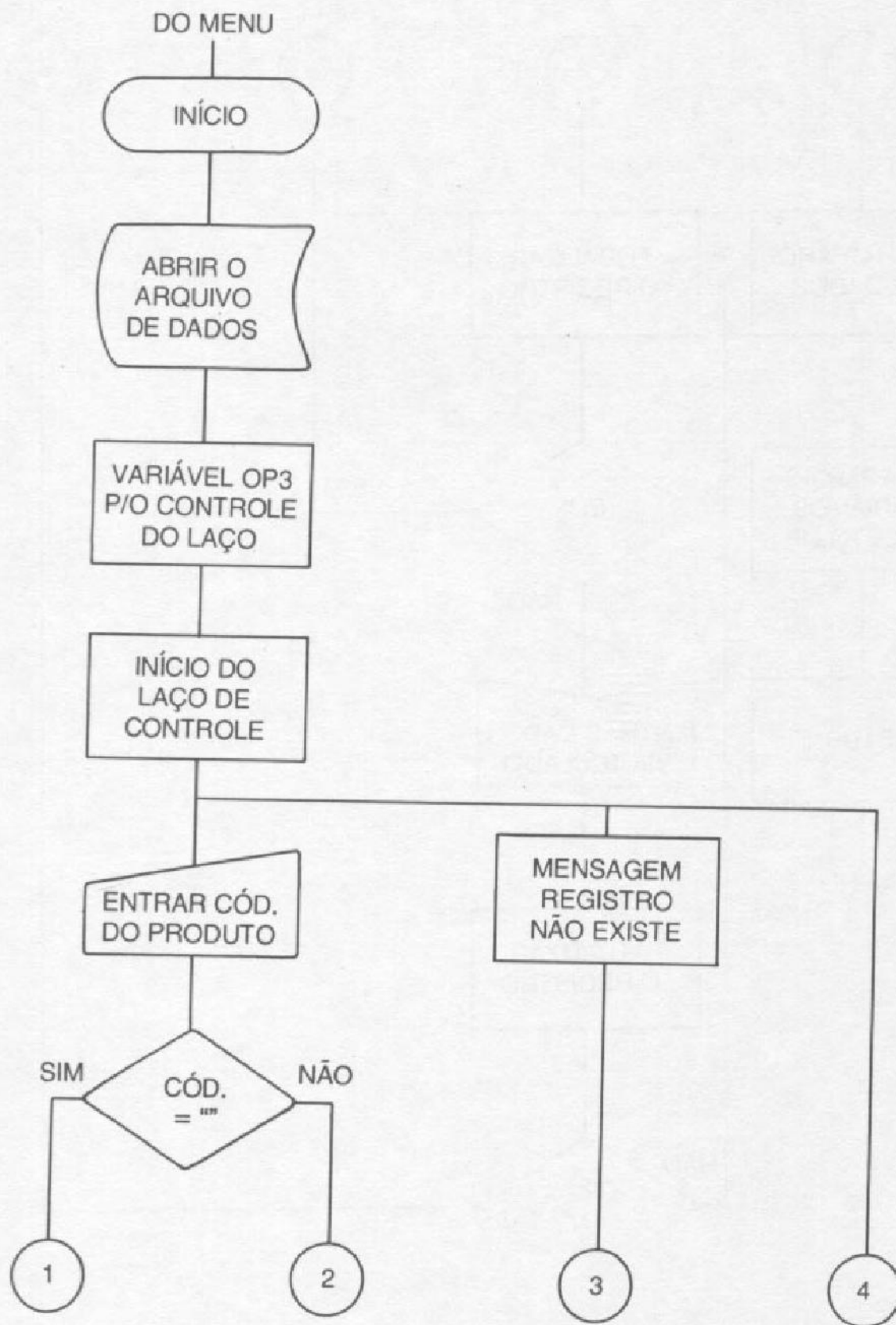
FLUXOGRAMA GERAL DO

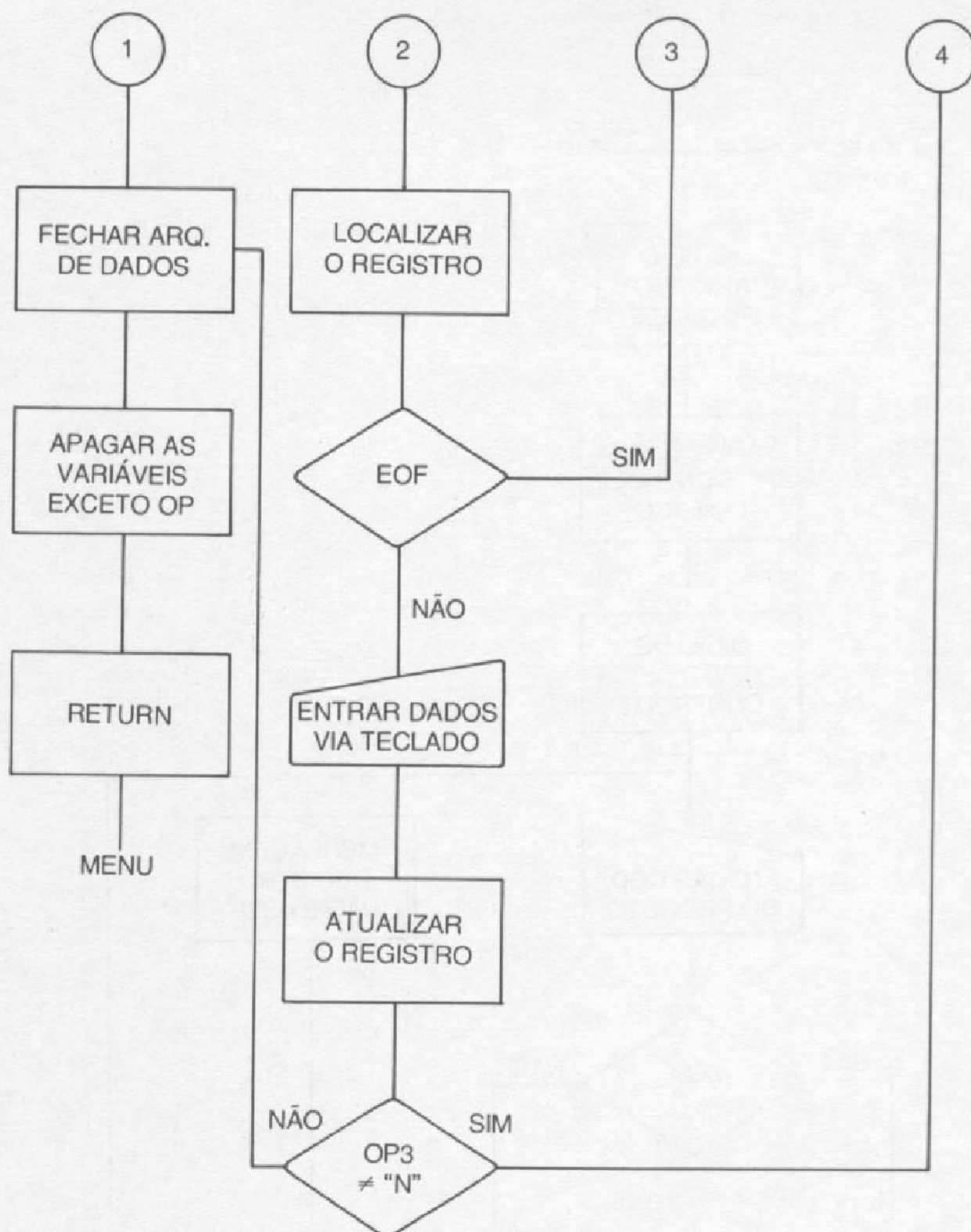
PROGRAMA "CONTROLE DE ESTOQUE"

FLUXOGRAMA DO MÓDULO SUPERVISOR - MENU

FLUXOGRAMA DO MóDULO SUPERVISOR - CONTINUAÇÃO

FLUXOGRAMA DO MÓDULO OPC-1 —

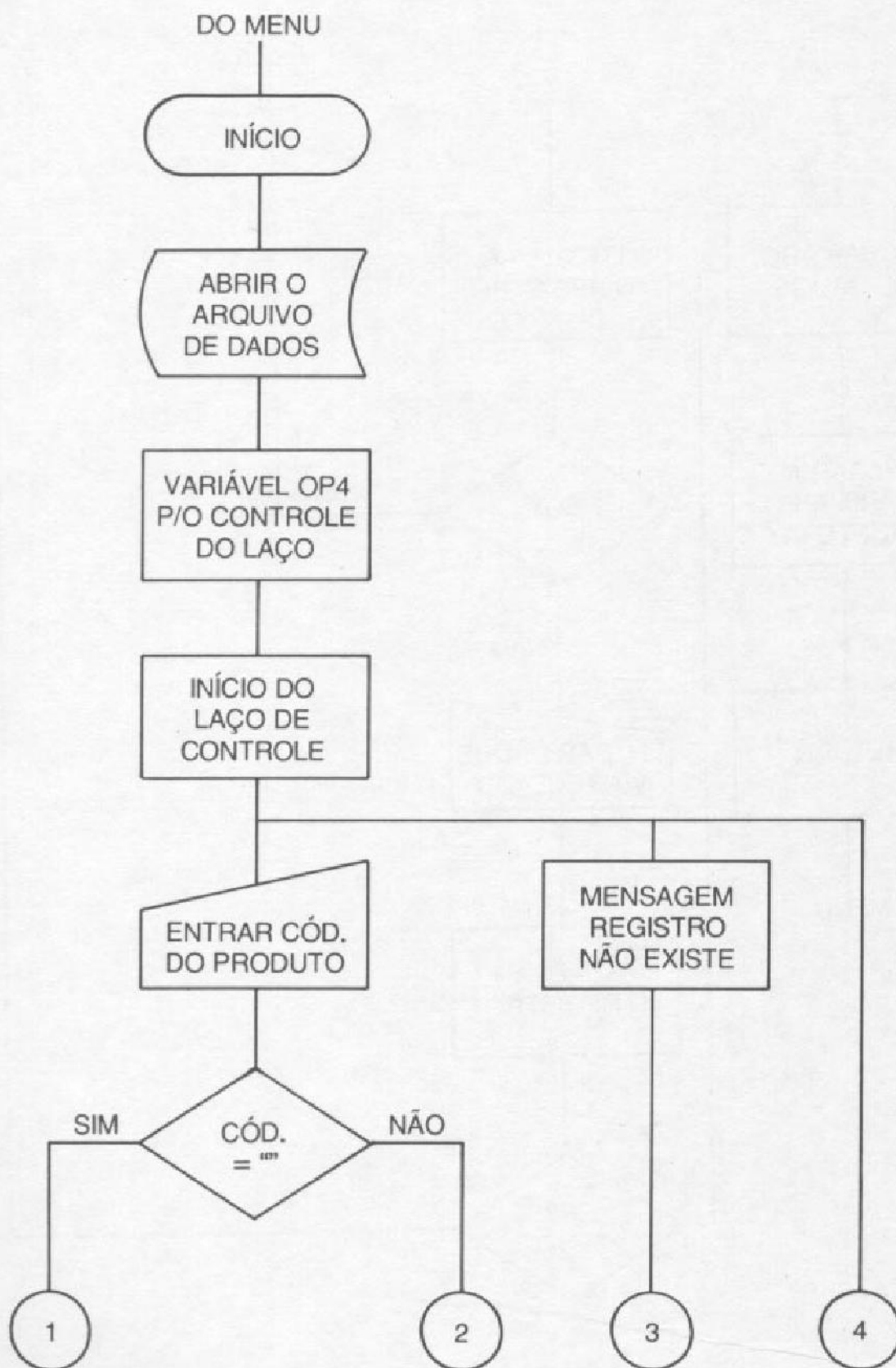
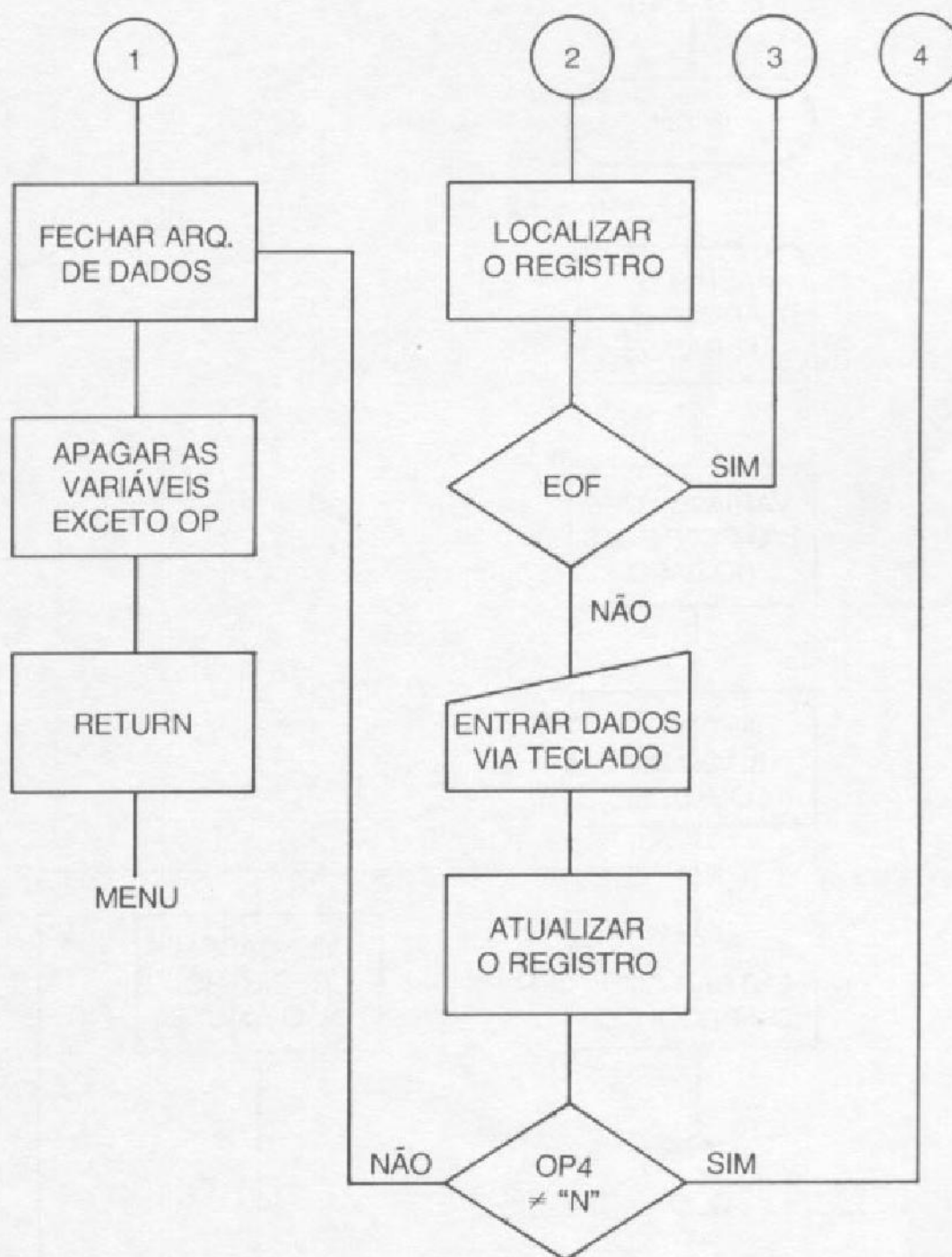
ENTRADA DE PRODUTO NOVO

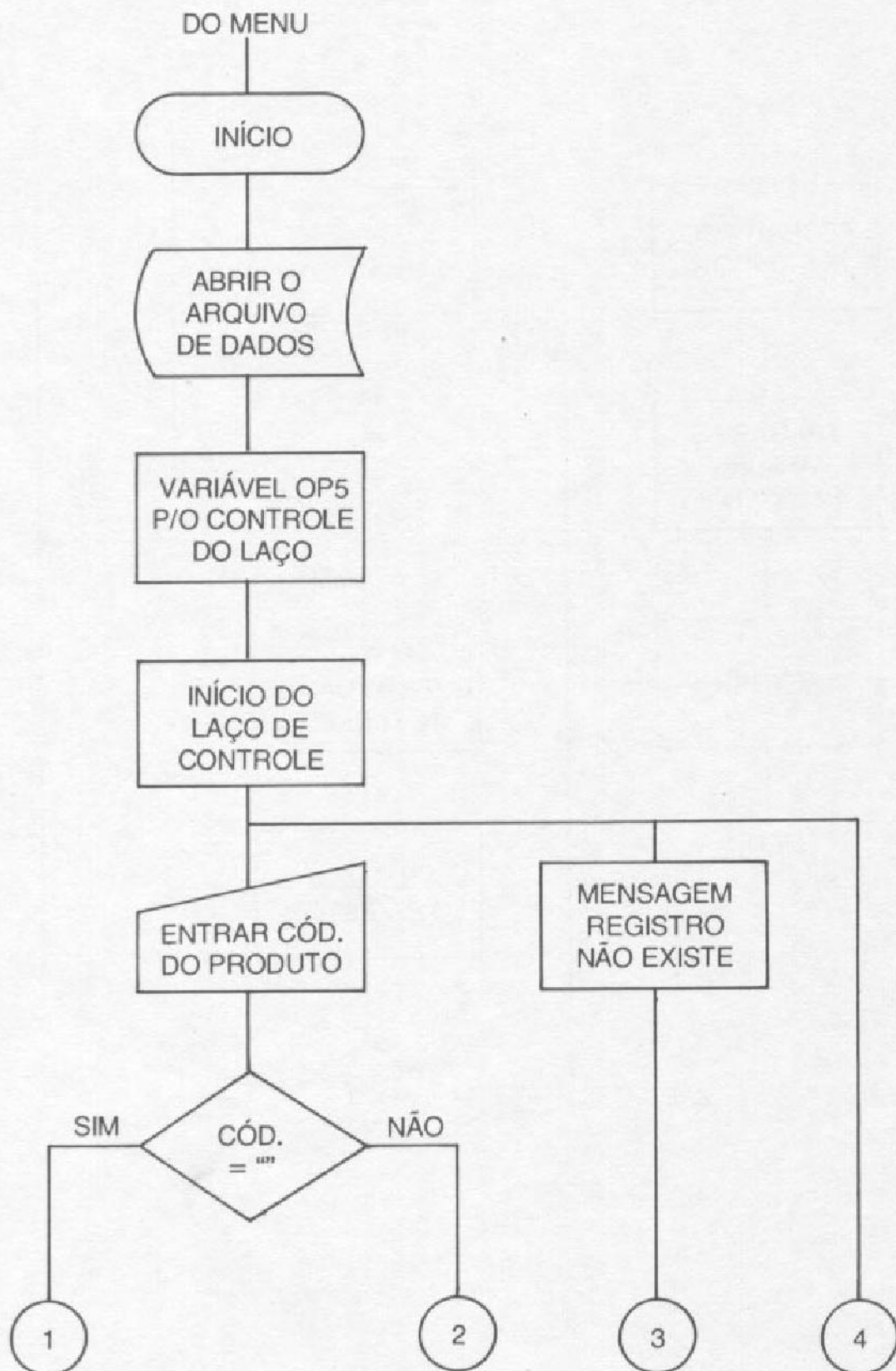
FLUXOGRAMA DO MÓDULO OPC-1 - CONTINUAÇÃO

FLUXOGRAMA DO MÓDULO OPC-2 -

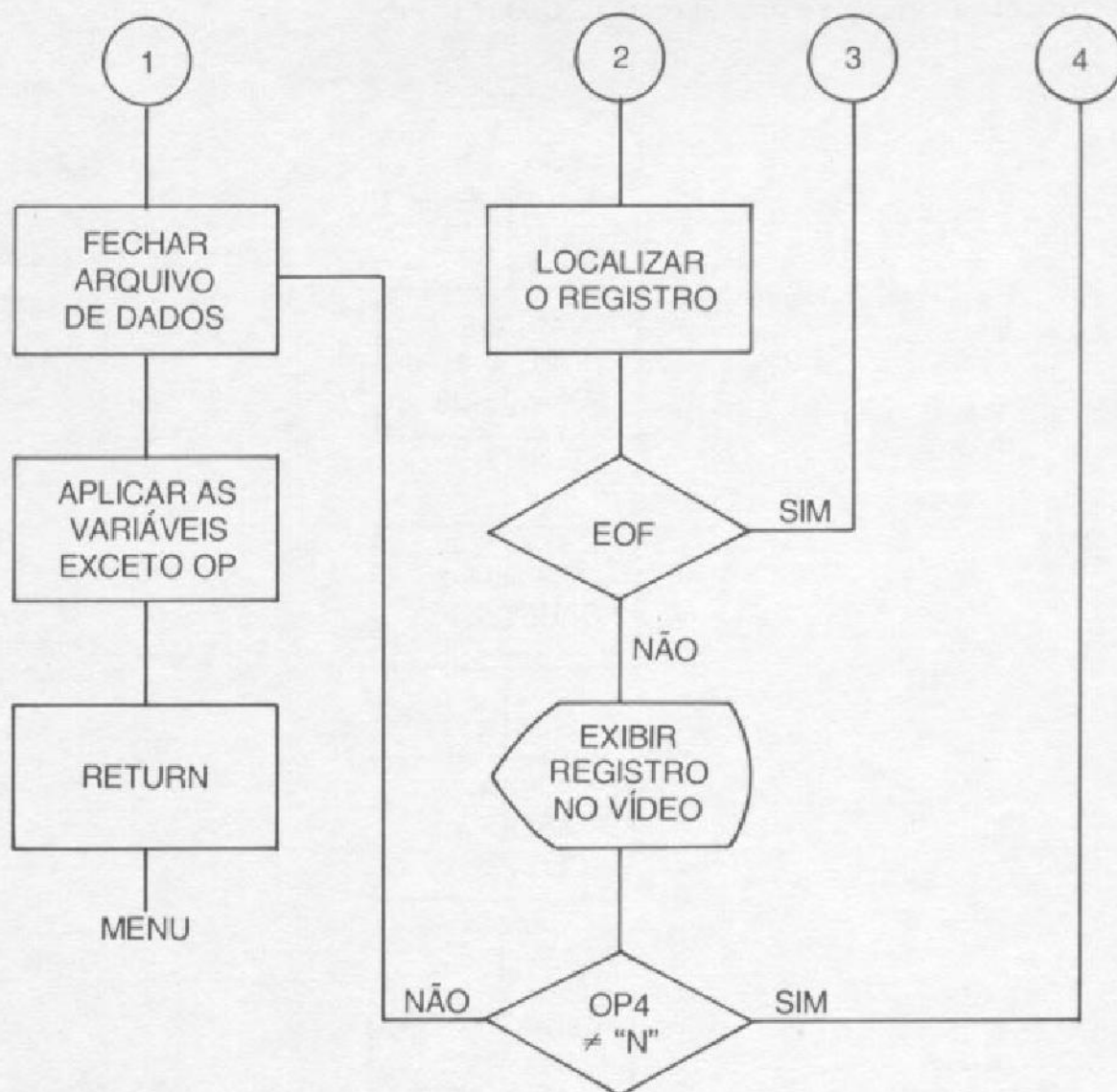
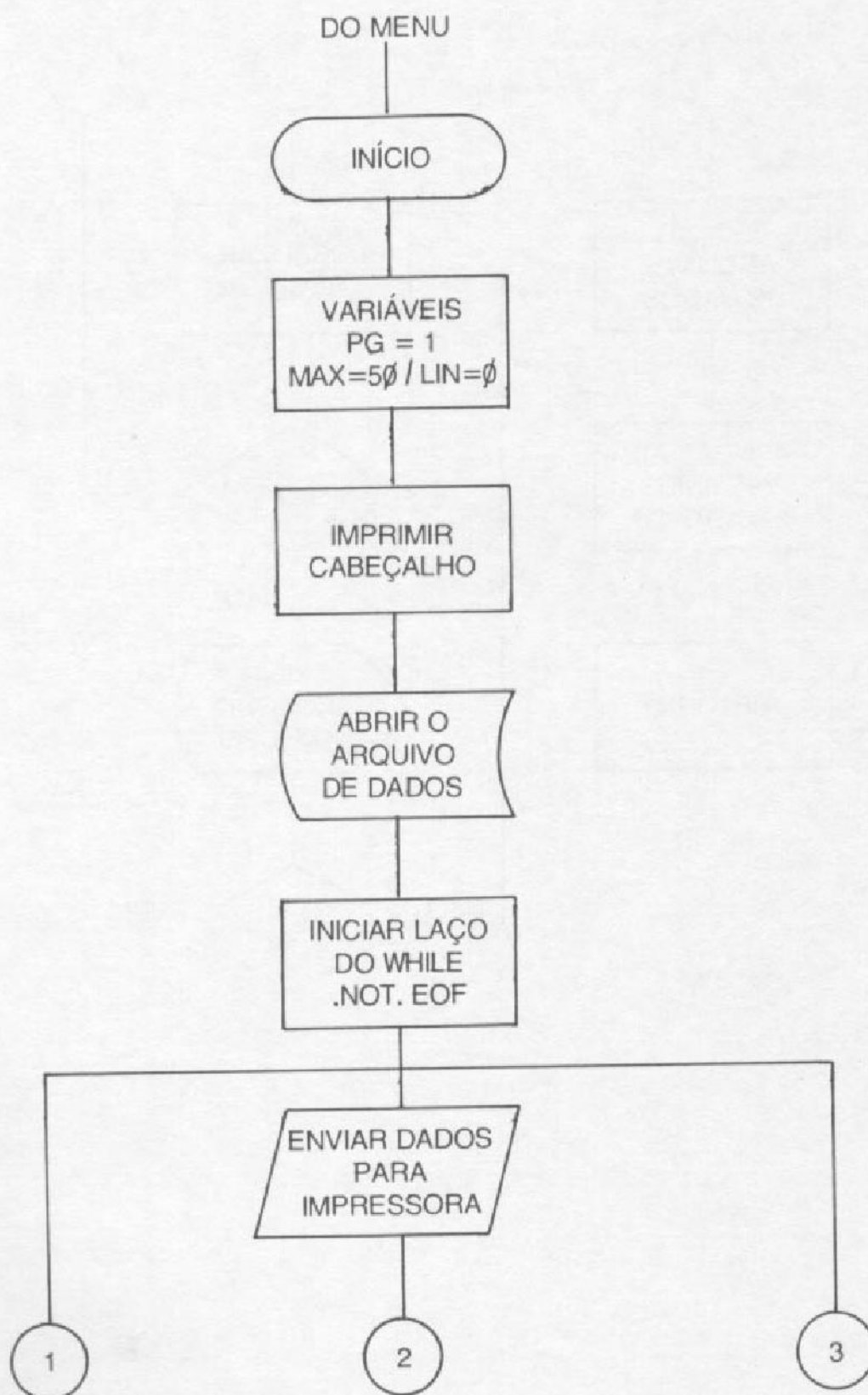
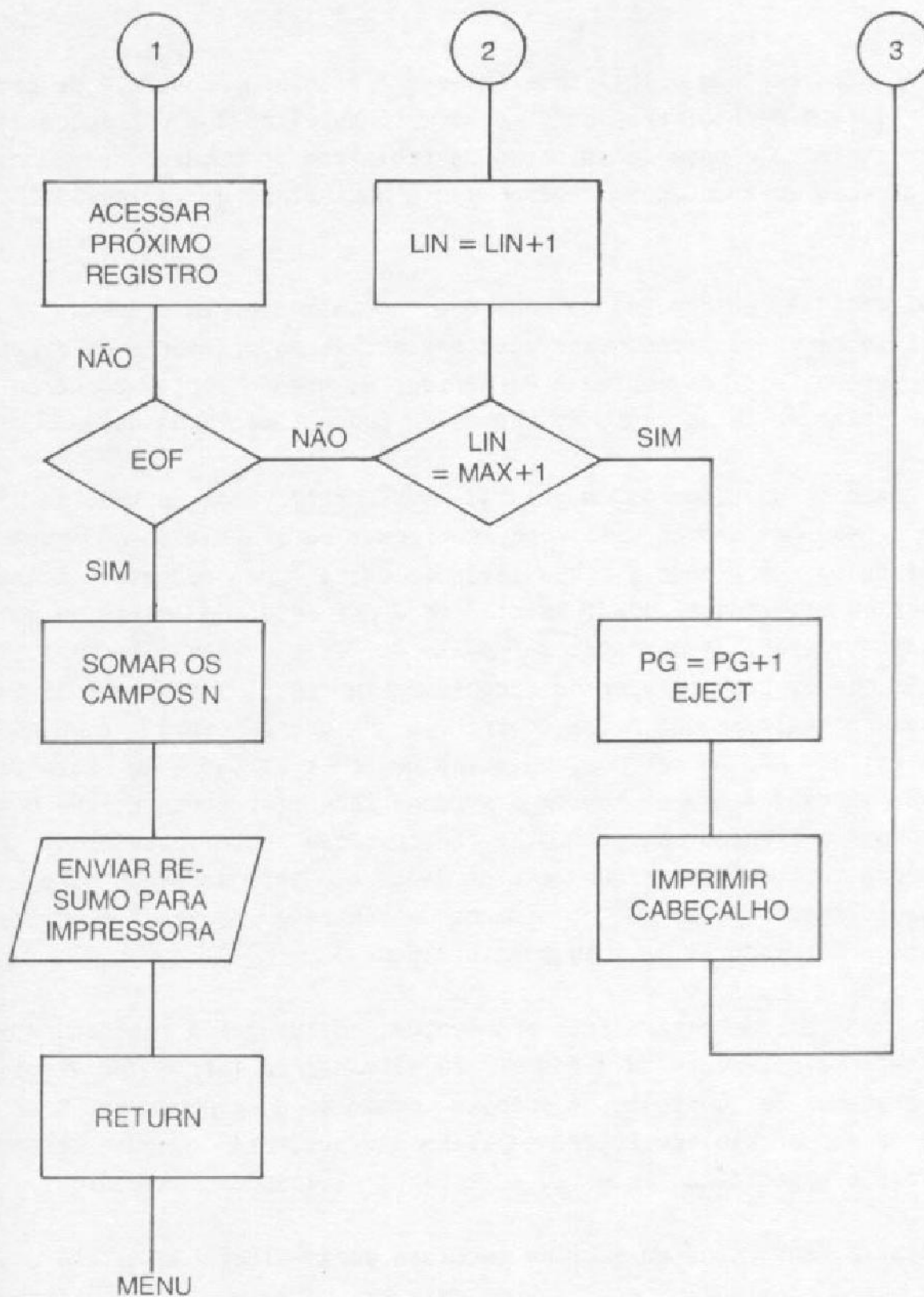
ENTRADA DE PRODUTO NÃO NOVO

FLUXOGRAMA DO MóDULO OPC-2 - CONTINUAÇÃO

FLUXOGRAMA DO MÓDULO OPC-3 -

SAÍDA DE PRODUTO

FLUXOGRAMA DO MÓDULO OPC-3 - CONTINUAÇÃO

FLUXOGRAMA DO MÓDULO OPC-4 –

DEVOLUÇÃO DE PRODUTO

FLUXOGRAMA DO MÓDULO OPC-4 - CONTINUAÇÃO

FLUXOGRAMA DO MóDULO OPC-5 -

PESQUISA DE PRODUTO

FLUXOGRAMA DO MÓDULO OPC-5 - CONTINUAÇÃO

FLUXOGRAMA DO MÓDULO OPC-6 -

EMISSÃO DE LISTAGEM

FLUXOGRAMA DO MÓDULO OPC-6 - CONTINUAÇÃO

10.9 – MÓDULOS COMPLEMENTARES AO PROGRAMA "CONTROLE DE ESTOQUE"

Propositalmente não incluímos em nosso programa "Controle de Estoque" recursos para alterações diretas e irrestritas dos conteúdos de qualquer registro e para eliminação de registros do arquivo de dados, os quais serão apresentados separadamente nos itens de índice 10.10, 10.11 e 10.12.

Tal exclusão se deve ao fato de que, normalmente, em pequenas empresas tais recursos ficam reservados somente ao proprietário da firma ou a seu preposto e, em empresas de médio e de grande porte, ao gerente administrativo ou ao chefe do centro de processamento de dados.

A razão de os recursos em questão serem reservados de modo a impedir o acesso aos mesmos pelo operador comum do sistema de processamento de dados se prende à necessidade de evitar que registros sejam alterados ou eliminados indevidamente, seja por descuido, má-fé ou outra razão qualquer, já que pode ser muito difícil, ou mesmo impossível em muitos casos, restabelecer ou recuperar informações adulteradas ou perdidas, se qualquer dos fatos ocorrer, ainda que percebido. E quando tais fatos não são percebidos, quem vai notar a falta de um registro eliminado indevidamente se também a mercadoria correspondente, que poderia chamar a atenção para o fato, "desapareceu misteriosamente"? E quem poderá ter certeza de que certos dados não estejam sendo alterados indevidamente, desde que o sistema de controle não esteja munido de recursos cerceadores de tais possibilidades?

Contudo, em que pesem tais argumentos, muitas vezes os recursos para alterar diretamente os dados e para eliminar registros são exigidos em sistemas de controle de estoque, mesmo porque, dependentemente do tipo de mercadoria envolvida, os itens são mutáveis ou suas características e especificações estão sujeitas a alterações freqüentes.

Todavia, nos casos em que os recursos para alteração direta dos dados e para eliminação de registros estejam normalmente disponíveis

no sistema, outras medidas para prevenção de erros, descuidos e má-fé devem ser implementadas no gerenciamento global dos dados.

São muitos os chamados "recursos de prevenção" que podem ser adotados em tais casos, mas são geralmente complexos e aplicáveis especificamente para cada tipo de arquivo e em função de suas condições gerais de funcionamento e de uso.

Introduziremos a seguir os dois módulos que possibilitam efetuar alterações diretas nos campos de dados e eliminar registros do arquivo. A critério do usuário ou do programador, esses módulos poderão ser incluídos no programa "Controle de Estoque" por nós desenvolvido neste Curso, bastando para tanto inserir as opções para sua execução no módulo MENU, respectivamente como opções de números 7 e 8, renumerando a de número 7 atual com o número 9.

O módulo introduzido a seguir é o que possibilita efetuar alterações diretamente nos conteúdos de campos de qualquer registro. Ele será apresentado em duas versões, a fim de oferecer ao usuário a escolha da que julgar mais conveniente. Uma utiliza os comandos GET e READ, enquanto a outra emprega o comando EDIT para a mesma finalidade.

10.10 - ELABORAÇÃO DO MÓDULO OPC-ALT - ALTERAÇÃO DE PRODUTO

Estando o dBASE ativado no microcomputador, comandar:

MODIFY COMMAND OPC-ALT <RETURN>

Em seguida, fazendo uso dos recursos do editor de programas acionado, digitar a listagem que segue:

```
*** PROGRAMA "CONTROLE DE ESTOQUE" ***
* MODULO OPC-ALT - ALTERAÇÃO DE PRODUTO
*
SET TALK OFF
```

```
ERASE
USE CTRLSTOQ
STORE " " TO OP7
DO WHILE !(OP7) <> "N"
@ 2,9 SAY "ALTERAÇÃO DE PRODUTO"
@ 3,9 SAY "===================="
@ 5,0 SAY "Digite o código ou <RETURN> p/encerrar."
@ 6,0
ACCEPT TO MCOD
   IF MCOD = " "
   USE
   RELEASE ALL EXCEPT OP
   RETURN
   ENDIF
LOCATE FOR CODIGO = !(MCOD)
   IF EOF
   @ 9,8 SAY "Código não encontrado!"
   ?
   ACCEPT "  Pressione <RETURN> para continuar." TO QQ
   RELEASE MCOD
   ERASE
   LOOP
   ENDIF
ERASE
@ 2,9 SAY "PRODUTO A ALTERAR:"
@ 5,5 SAY "Código " GET CODIGO
@ 7,5 SAY "Produto " GET DENOM
@ 9,5 SAY "Departamento " GET DEP
@ 10,5 SAY "Estoque atual " GET QTAT
@ 11,5 SAY "Estoque mínimo " GET QTMIN
@ 12,5 SAY "Preço de custo " GET PRC
@ 13,5 SAY "Preço de venda " GET PRV
@ 15,0 SAY "Valor de custo do estoque:"
@ 15,28 GET VALC PICTURE "#######.##"
@ 16,0 SAY "Valor de venda do estoque:"
@ 16,28 GET VALV PICTURE "#######.##"
READ
@ 18,0
?
```

```
ACCEPT "Outra alteração? (<RETURN>=SIM, <N>=NÃO)" TO OP7
ERASE
ENDDO
RELEASE ALL EXCEPT OP
USE
RETURN
```

Terminada e conferida a digitação, pressionar as teclas CONTROL e W simultaneamente a fim de encerrar a edição do módulo e fazer com que seja gravado automaticamente no disquete.

COMENTÁRIOS SOBRE O MÓDULO OPC-ALT:

Este módulo é semelhante ao módulo OPC-5 - PESQUISA DE PRODUTO. A diferença principal entre ambos é que neste não é utilizado o comando CLEAR GETS, de modo que todos os conteúdos de campos expostos através do comando GET são abrangidos pelo comando READ na modalidade de edição, permitindo efetuar as alterações desejadas diretamente sobre esses mesmos conteúdos, característica essa do comando READ já estudada.

De resto, os comandos utilizados são os mesmos empregados no módulo OPC-5, tendo sido já comentados.

A listagem apresentada a seguir refere-se à segunda versão do módulo para alteração direta dos conteúdos de campos de registro.

10.11 - ELABORAÇÃO DO MÓDULO OPC-EDIT - EDIÇÃO DE PRODUTO

Com o dBASE ativado no microcomputador, comandar:

```
MODIFY COMMAND OPC-EDIT
```

Digitar em seguida a listagem que segue:

```
*** PROGRAMA "CONTROLE DE ESTOQUE" ***
* MóDULO OPC-EDIT - EDIÇÃO DE PRODUTO
*
SET TALK OFF
ERASE
USE CTRLSTOQ
STORE " " TO OP7
DO WHILE !(OP7) <> "N"
@ 2,9 SAY "EDIÇÃO  DE  PRODUTO"
@ 3,9 SAY "==================="
@ 5,0 SAY "Digite o código ou <RETURN> p/encerrar."
@ 6,0
ACCEPT TO MCOD
   IF MCOD = " "
   USE
   RELEASE ALL EXCEPT OP
   RETURN
   ENDIF
LOCATE FOR CODIGO = !(MCOD)
   IF EOF
   @ 9,8 SAY "Código não encontrado!"
   ?
   ACCEPT "  Pressione <RETURN> para continuar." TO QQ
   RELEASE MCOD
   ERASE
   LOOP
   ENDIF
ERASE
@ 10,2 SAY "Após fazer as correções tecle CTRL+W"
?
EDIT #
ERASE
@ 10,0
ACCEPT "Outro produto? (<RETURN>=SIM, <N>=NÃO)" TO OP7
ERASE
ENDDO
RELEASE ALL EXCEPT OP
USE
RETURN
```

Terminada e conferida a digitação, pressionar a uma só vez as teclas CONTROL e W a fim de encerrar a edição do módulo e fazer com que seja gravado automaticamente no disquete.

COMENTÁRIOS SOBRE O MÓDULO OPC-EDIT:

Este módulo gera a exibição de todos os campos de registro e respectivos conteúdos em modo de edição plena, em conformidade com as características próprias do comando EDIT, permitindo que alterações sejam efetuadas diretamente sobre esses mesmos conteúdos.

Os demais comandos utilizados neste módulo já foram comentados em relação aos módulos anteriormente apresentados.

A seguir é apresentado o módulo que possibilita eliminar registros do arquivo.

10.12 - ELABORAÇÃO DO MÓDULO OPC-DEL - ELIMINAÇÃO DE PRODUTO

Com o dBASE ativado no microcomputador, comandar:

MODIFY COMMAND OPC-DEL <RETURN>

e digitar após a listagem que segue:

```
*** PROGRAMA "CONTROLE DE ESTOQUE" ***
* MÓDULO OPC-DEL - ELIMINAÇÃO DE PRODUTO
*
SET TALK OFF
ERASE
USE CTRLSTOQ
STORE " " TO OP8
DO WHILE !(OP8) <> "N"
```

```
@ 2,9 SAY "ELIMINAÇÃO DE PRODUTO"
@ 3,9 SAY "====================="
@ 5,0 SAY "Digite o código ou <RETURN> p/encerrar."
@ 6,0
ACCEPT TO MCOD
   IF MCOD = " "
   USE
   RELEASE ALL EXCEPT OP
   RETURN
   ENDIF
LOCATE FOR CODIGO = !(MCOD)
   IF EOF
   @ 9,8 SAY "Código não encontrado!"
   ?
   ACCEPT "  Pressione <RETURN> para continuar." TO QQ
   RELEASE MCOD
   ERASE
   LOOP
   ENDIF
ERASE
@ 2,9 SAY "PRODUTO A ELIMINAR:"
@ 5,5 SAY "Código " GET CODIGO
@ 7,5 SAY "Produto " GET DENOM
@ 9,5 SAY "Departamento " GET DEP
@ 10,5 SAY "Estoque atual " GET QTAT
@ 11,5 SAY "Estoque mínimo " GET QTMIN
@ 12,5 SAY "Preço de custo " GET PRC
@ 13,5 SAY "Preço de venda " GET PRV
CLEAR GETS
@ 15,0 SAY "Valor de custo do estoque:"
@ 15,28 SAY VALC USING "#,###,###.##"
@ 16,0 SAY "Valor de venda do estoque:"
@ 16,28 SAY VALV USING "#,###,###.##"
?
ACCEPT "      Confirma a eliminação? (S/N)" TO RESP
   IF !(RESP) = "S"
   DELETE
   ?
   ? "  Registro assinalado para eliminação."
```

```
   ? "  Será eliminado após o processamento."
   ENDIF
?
ACCEPT "Outro produto? (<RETURN>=SIM, <N>=NÃO)" TO OPB
ERASE
ENDDO
RELEASE ALL EXCEPT OP
PACK
USE
RETURN
```

Terminada e conferida a digitação, pressionar as teclas CONTROL e W para encerrar a edição do módulo e provocar a sua gravação automática no disquete.

COMENTÁRIOS SOBRE O MÓDULO OPC-DEL:

Este módulo gera a exibição no vídeo de todo o conteúdo do registro de arquivo selecionado, oferecendo ao usuário a opção de eliminá-lo do arquivo, o que é feito através do comando DELETE, no seu modo de funcionamento normal. Ao ser encerrado o processamento pelo módulo, o comando PACK é automaticamente acionado a fim de confirmar a exclusão do(s) registro(s) assinalado(s). Caso seja pretendido apenas assinalar os registros para deleção, sem confirmá-la, o comando PACK deverá ser excluído do módulo.

Para os módulos introduzidos nos itens 10.10, 10.11 e 10.12 não foram elaborados fluxogramas, já que sua estruturação geral se assemelha aos módulos apresentados precedentemente para o mesmo programa.

11 — COMANDOS DA SÉRIE PLUS

Os comandos da série PLUS, incorporados ao dBASE mais recentemente, permitem introduzir em programas elaborados neste sistema alguns dos recursos mais avançados da informática.

Graças a tais comandos, a automatização das seguintes facilidades pode ser introduzida no gerenciamento de bancos de dados através do sistema dBASE:

- teste de validade de data;
- determinação de intervalo (em dias) entre duas datas;
- determinação de data futura com base em acréscimo de prazo (em dias) sobre data inicial, para efeito de vencimento de contratos, duplicatas etc,;
- determinação de data futura, com transferência automática para segunda-feira quando a data coincide com sábado ou domingo;
- conversão da pontuação do sistema americano - inerente nos computadores - para o sistema europeu, que é o adotado no Sistema Métrico Decimal do Brasil;
- troca de brancos à esquerda de números por zeros;

- criação de dígito de controle para número, de acordo com o sistema "módulo 11";

- teste de validade de CGC;

- teste de validade de CPF;

- conversão de números inteiros simples para o padrão monetário do sistema europeu, que é o adotado no Brasil;

- criação e uso de senha para abertura de programas ou arquivos.

Os comandos da série PLUS serão descritos a seguir. Cada comando será acompanhado por um pequeno programa para testá-lo e que servirá, ao mesmo tempo, como módulo-tipo para adoção em programas efetivos.

Comando PLUS CDATE

*** PLUS CDATE testa a validade de uma data. Sua sintaxe é:

PLUS CDATE <variável1> TO <variável2>

Variável1 deve ser uma variável alfanumérica, ou tipo C, na qual deve ser armazenada a data no formato DDMMAA (dia, mês e ano, todos com dois dígitos. Exemplo: 160688).

Variável2 deve ser uma variável alfanumérica na qual deve ser armazenado previamente, isto é, antes de ser executado o comando em foco, um espaço em branco (" ").

Quando a data testada é válida, PLUS CDATE atribui um 1 à Variável2. Em caso contrário será atribuído um 0 à essa variável. Assim, a validade ou não da data testada é constatada através da verificação do conteúdo da Variável2.

O pequeno programa apresentado a seguir testa o comando descrito e exemplifica o seu uso:

PROGRAMA-TESTE "PLUS CDATE"

```
SET TALK OFF
ERASE
STORE " " TO CT
DO WHILE CT = " "
?
?
? "Digite a data no formato DDMMAA "
? "      (Tecle ESC para parar.)"
?
ACCEPT TO DATA
STORE " " TO VER
PLUS CDATE DATA TO VER
   IF VER = "1"
   ?
   ? "    DATA OK!"
   ELSE
   ?
   ? "    DATA INVÁLIDA!"
   ENDIF
ENDDO
RELEASE ALL
SET TALK ON
CANCEL
```

Comando PLUS DDATE

*** PLUS DDATE calcula o intervalo entre duas datas e retorna o resultado em quantidade de dias. Sua sintaxe é:

PLUS DDATE <variável1> <variável2>
TO <variável3>

Variável1 deve ser uma variável alfanumérica (tipo C ou "string") na qual deve ser armazenada a data inicial no formato DDMMAA (dia, mês e ano, todos com dois dígitos. Exemplo: 050788).

Variável2 deve ser também uma variável alfanumérica na qual deve ser armazenada a data final no formato DDMMAA.

Variável3 deve ser igualmente uma variável alfanumérica na qual devem ser armazenados cinco espaços em branco ("   ") antes de ser executado o comando PLUS DDATE.

O resultado do cálculo efetuado pelo comando PLUS DDATE é armazenado na Variável3, de modo que a verificação de seu conteúdo fornece o intervalo (em dias) existente entre as duas datas processadas.

O pequeno programa que segue testa o comando descrito e exemplifica o seu uso:

PROGRAMA-TESTE "PLUS DDATE"

```
* TESTE "PLUS DDATE"
SET TALK OFF
ERASE
STORE " " TO CT
DO WHILE CT = " "
?
?
? "Digite a data inicial"
? " no formato DDMMAA ou"
ACCEPT "    tecle ESC p/parar " TO DATANT
?
ACCEPT "Digite  a data  final " TO DATPOS
STORE "      " TO DIAS
PLUS DDATE DATANT DATPOS TO DIAS
?
? "   DIAS DECORRIDOS:", DIAS
?
ENDDO
```

```
RELEASE ALL
SET TALK ON
CANCEL
```

Comando PLUS ADATE

*** PLUS ADATE determina uma data futura com base em acréscimo de até 242 dias sobre uma data indicada. Sua sintaxe é:

```
PLUS ADATE <variável1> <variável2>
           <variável3> TO <variável4>
```

Variável1 deve ser uma variável alfanumérica na qual deve ser armazenada a data inicial no formato DDMMAA.

Variável2 deve ser também uma variável alfanumérica na qual devem ser armazenados os dígitos correspondentes à quantidade de dias a ser acrescentada à data inicial. Tal quantidade de dias deve ser digitada sempre com três dígitos, conforme um dos formatos indicados a seguir: 00X, 0XX, XXX, sendo X a quantidade desejada.

Variável3 deve ser igualmente uma variável alfanumérica na qual deve ser armazenado um 0 ou um 1, a fim de determinar o critério para contagem dos dias indicados. A contagem será feita a partir da data inicial se for atribuído um 0 a essa variável. Se lhe for atribuído um 1 a contagem será feita a partir do primeiro dia do mês seguinte à data inicial.

Variável4 deve ser também uma variável alfanumérica na qual devem ser armazenados seis espaços em branco ("   ") antes de ser executado o comando PLUS ADATE.

Depois de executado o comando, o resultado será armazenado na Variável4, de modo que a leitura de seu conteúdo indicará a data futura.

O pequeno programa listado a seguir se presta a testar o comando ora descrito, exemplificando ao mesmo tempo o seu uso:

PROGRAMA-TESTE "PLUS ADATE"

```
* TESTE "PLUS ADATE"
SET TALK OFF
ERASE
STORE " " TO CT
DO WHILE CT = " "
?
?
? "Digite a data inicial no forma-"
ACCEPT " to DDMMAA ou tecle ESC p/parar " TO DATA1
?
? "Digite no  formato 00X, 0XX ou XXX a"
ACCEPT " quantidade de  dias a acrescentar " TO DIAS
?
? "Contagem a partir da data inicial ou"
ACCEPT " primeiro dia do  mês seguinte (I/S) " TO T
STORE !(T) TO T1
   IF T1 = "I"
   STORE "0" TO INI
   ELSE
   STORE "1" TO INI
   ENDIF
STORE "        " TO DATA2
PLUS ADATE DATA1 DIAS INI TO DATA2
?
? " *** A PRÓXIMA DATA É ", DATA2
ENDDO
SET TALK ON
CANCEL
```

Comando PLUS TDATE

*** PLUS TDATE determina uma data futura com base em acréscimo de dias sobre uma data indicada, transferindo para a SEGUNDA-FEIRA a

DATA FUTURA se esta coincidir com SÁBADO ou DOMINGO.

Sua sintaxe é:

```
PLUS TDATE <variável1> <variável2>
                <variável3> TO <variável4>
```

Basicamente o comando PLUS TDATE é similar ao comando PLUS ADATE, sendo ambos regidos pela mesma sintaxe.

Aplicam-se a este comando todas as explanações feitas sobre o seu similar.

Testes com o comando PLUS TDATE podem ser feitos com o programa cuja listagem é apresentada a seguir:

PROGRAMA-TESTE "PLUS TDATE"

```
* TESTE "PLUS TDATE"
SET TALK OFF
ERASE
STORE " " TO CT
DO WHILE CT = " "
?
?
? "Digite a data inicial no forma-"
ACCEPT " to DDMMAA ou tecle ESC p/parar " TO DATA1
?
? "Digite no  formato 00X, 0XX ou XXX a"
ACCEPT " quantidade de  dias a acrescentar " TO DIAS
?
? "Contagem a partir da data inicial ou"
ACCEPT " primeiro dia do  mês seguinte (I/S) " TO T
STORE !(T) TO T1
   IF T1 = "I"
   STORE "0" TO INI
   ELSE
   STORE "1" TO INI
   ENDIF
```

```
STORE "        " TO DATA2
PLUS TDATE DATA1 DIAS INI TO DATA2
?
? " *** A PRÓXIMA DATA É ", DATA2
ENDDO
SET TALK ON
CANCEL
```

Comando PLUS CHANGE

*** PLUS CHANGE converte a pontuação do sistema monetário americano para o europeu, que é o adotado no Brasil. Sua sintaxe é:

PLUS CHANGE <variável1> TO <variável2>

Variável1 deve ser uma variável alfanumérica na qual devem ser armazenados os dígitos correspondentes ao valor, no formato do sistema americano, isto é, com ponto separando os centavos e vírgulas separando cada três dígitos da parte de inteiros.

Variável2 deve ser também criada e definida como variável alfanumérica e com a mesma quantidade de espaços da Variável1, antes de ser executado o comando PLUS CHANGE.

Após a execução de PLUS CHANGE, a conversão feita é armazenada na Variável2 e obtida através da leitura do conteúdo dessa variável.

Deve ser observado que os valores convertidos pelo comando em foco não são aritmeticamente válidos no dBASE, pois são conteúdos de variáveis alfanuméricas. Como sabemos, a computação de números ou valores só pode ser feita diretamente através de dígitos números ou através de variáveis numéricas. Assim, independentemente do processamento através do comando PLUS CHANGE - cuja finalidade é a conversão de formato apenas para efeito de clareza e estética na apresentação dos dados -, os valores devem ser atribuídos a variáveis numéricas e computados paralelamente através das mesmas para fins aritméticos. No Capítulo 12 - RECURSOS SOFISTICADOS - é apresentado um módulo que permite computar efetivamente valores aritméticos, ou monetários, e convertê-

los no padrão de pontuação europeu para efeito de exibição no vídeo ou reprodução em relatórios emitidos através de impressora.

A seguir é listado um pequeno programa para teste do comando PLUS CHANGE, servindo o mesmo como exemplo para programas efetivos:

PROGRAMA-TESTE "PLUS CHANGE"

```
* TESTE "PLUS CHANGE"
SET TALK OFF
ERASE
STORE "S" TO RESP
DO WHILE RESP = "S"
@ 12,0 SAY "...
STORE "                    " TO VAL
?
@ 6,0 SAY "Qual o valor? "GET VAL PICTURE "#,###,###.##"
READ
STORE VAL TO VALOR1, VALOR2
PLUS CHANGE VALOR1 TO VALOR2
@ 7,0
? "Sua conversão: ",VALOR2
?
?
ACCEPT "Outro teste? (S/N)" TO RESP
STORE !(RESP) TO RESP
ENDDO
RELEASE ALL
SET TALK ON
CANCEL
```

Comando PLUS LZERO

*** PLUS LZERO troca brancos à esquerda de números por zeros.

Sua sintaxe é:

PLUS LZERO <variável1> TO <variável2>

Variável1 deve ser uma variável alfanumérica que deve conter, com brancos à esquerda, os dígitos referentes ao número a ser processado.

Variável2 deve ser também uma variável alfanumérica com a mesma quantidade de espaços da Variável1. Deve ser criada e definida antes da execução do comando PLUS LZERO.

Depois da execução do comando PLUS LZERO, o número convertido é armazenado na Variável2, da qual é obtido.

Os números convertidos por esse comando não são computáveis, pois não são aritmeticamente válidos no dBASE, já que são representados por variáveis alfanuméricas. Como sabemos, a computação de números ou valores só pode ser feita diretamente ou através de variáveis numéricas, de modo que, independentemente do processamento por PLUS LZERO, os números devem ser atribuídos a variáveis numéricas e computados através das mesmas para fins de cálculos aritméticos.

No Capítulo 12 - RECURSOS SOFISTICADOS - é apresentado um módulo que permite processar números aritmeticamente e convertê-los paralelamente com zeros à esquerda.

O pequeno programa apresentado a seguir testa o comando descrito e exemplifica o seu uso:

PROGRAMA-TESTE "PLUS LZERO"

```
* TESTE "PLUS LZERO"
SET TALK OFF
ERASE
STORE " " TO CT
DO WHILE CT = " "
?
?
? "Digite um nº ou"
INPUT " tecle ESC p/parar  " TO NUM
```

```
STORE STR(NUM,10,0) TO NUM1
STORE "           " TO NUM2
PLUS LZERO NUM1 TO NUM2
?
? "      Número convertido: ", NUM2
ENDDO
RELEASE ALL
SET TALK ON
CANCEL
```

Comando PLUS MOD11

*** PLUS MOD11 calcula e fornece o dígito de controle de um número, de acordo com o padrão "módulo 11". Sua sintaxe é:

PLUS MOD11 (variável1) TO (variável2)

Variável1 deve ser uma variável alfanumérica na qual deve ser armazenado o número que receberá o dígito de controle.

Variável2 deve ser também uma variável alfanumérica na qual deve ser armazenado apenas um espaço em branco (" "). Deve ser criada e definida antes da execução do comando PLUS MOD11.

Depois de gerado pelo comando PLUS MOD11, o dígito de controle é armazenado na Variável2, da qual é obtido.

O pequeno programa listado a seguir testa o comando descrito, ao mesmo tempo que exemplifica o seu uso:

PROGRAMA-TESTE "PLUS MOD11"

```
* TESTE "PLUS MOD11"
SET TALK OFF
ERASE
STORE " " TO CT
DO WHILE CT = " "
```

```
?
?
? "Digite um nº ou"
INPUT " tecle ESC p/parar  " TO NUM
STORE STR(NUM,10,0) TO NUM1
STORE " " TO DIGIT
PLUS MOD11 NUM1 TO DIGIT
?
? "      O nº com dígito é ",NUM1 + "-" + DIGIT
ENDDO
RELEASE ALL
SET TALK ON
CANCEL
```

Comando PLUS CGC

*** O comando PLUS CGC testa a validade do número de CGC (Cadastro Geral de Contribuintes, do Ministério da Fazenda). Sua sintaxe é:

PLUS CGC <variável1> TO <variável2>

Variável1 deve ser uma variável alfanumérica na qual os dígitos do CGC a ser conferido devem ser armazenados.

Variável2 deve ser também uma variável alfanumérica na qual deve ser armazenado apenas um espaço em branco (" ") ou qualquer caractere. Deve ser criada e definida antes da execução do comando PLUS CGC.

Após a execução do comando PLUS CGC, o dBASE atribui à Variável2 um 1 ou um 0. Se o número do CGC for válido, será atribuído um 1 a essa variável. Ser-lhe-á atribuído um 0 em caso contrário.

Assim, a validade ou não do número de CGC testado é deduzida pela interpretação do conteúdo da Variável2, após a execução de PLUS CGC.

O pequeno programa apresentado a seguir serve para testar o comando descrito e exemplica o seu uso:

PROGRAMA-TESTE "PLUS CGC"

```
* TESTE "PLUS CGC"
SET TALK OFF
ERASE
STORE " " TO CT
DO WHILE CT = " "
STORE " " TO REP
?
?
? "Digite o CGC sem pontos nem espaços"
? "      em branco ou tecle ESC p/parar"
?
ACCEPT TO MCGC
PLUS CGC MCGC TO REP
   IF REP = "1"
   ?
   ? "     O nº do CGC está correto!"
   ELSE
   ?
   ? "     O nº do CGC está errado!"
   ENDIF
ENDDO
RELEASE ALL
SET TALK ON
CANCEL
```

Comando PLUS CPF

*** PLUS CPF testa a validade do número de CPF (Cadastro de Pessoa Física, do Ministério da Fazenda). Sua sintaxe é:

PLUS CPF <variável1> TO <variável2>

Basicamente, as características deste comando são idênticas às do

comando PLUS CGC, sendo ambos regidos pela mesma sintaxe.

Aplicam-se a PLUS CPF todas as explanações feitas sobre PLUS CGC.

O pequeno programa apresentado a seguir serve para testes do comando e também exemplifica o seu uso:

PROGRAMA-TESTE "PLUS CPF"

```
* TESTE "PLUS CPF"
SET TALK OFF
ERASE
STORE " " TO CT
DO WHILE CT = " "
STORE " " TO REP
?
?
? "Digite o CPF sem pontos nem espaços"
? "      em branco ou tecle ESC p/parar"
?
ACCEPT TO MCPF
PLUS CPF MCPF TO REP
   IF REP = "1"
   ?
   ? "   O nº do CPF está correto!"
   ELSE
   ?
   ? "   O nº do CPF está errado!"
   ENDIF
ENDDO
RELEASE ALL
SET TALK ON
CANCEL
```

Comando PLUS FORMAT

*** PLUS FORMAT converte números inteiros simples para o padrão mone-

tário do sistema europeu, que é o adotado no Brasil.

Sua sintaxe é:

```
PLUS FORMAT <variável1> <variável2>
                        TO <variável3>
```

Variável1 deve ser uma variável alfanumérica na qual devem ser armazenados os dígitos correspondentes ao número a ser convertido para o padrão monetário. Os dígitos não devem ser separados por ponto decimal nem por vírgulas a cada três casas da parte inteira. Por exemplo, um valor igual a 12,345.60 (ou 12.345,60 no formato brasileiro) deve ser digitado no formato 1234560.

Variável2 deve ser também uma variável alfanumérica à qual deve deve ser atribuído o dígito correspondente à quantidade de casas decimais desejadas para o número convertido.

Variável3 deve ser igualmente uma variável alfanumérica com espaços em branco em quantidade igual à correspondente aos espaços ocupados pela Variável1, ou maior, e mais os espaços destinados aos pontos de separação de cada três casas da parte inteira e a vírgula de separação das casas decimais. Esta variável deve ser criada e definida antes da execução do comando PLUS FORMAT.

O número já convertido por PLUS FORMAT é armazenado na Variável3, de onde deve ser recuperado para exibição ou reprodução impressa.

O valor convertido por PLUS FORMAT não é computável aritmeticamente, em vista de ser conteúdo de variável alfanumérica, do mesmo modo que ocorre com os comandos PLUS CHANGE e PLUS LZERO. Assim, para que o valor efetivo seja computado aritmeticamente, deve ser atribuído a uma variável numérica e processado independentemente, ainda que paralelamente. No Capítulo 12 - RECURSOS SOFISTICADOS - são apresentados módulos que utilizam o processamento paralelo.

O pequeno programa apresentado a seguir exemplifica o uso do comando PLUS FORMAT e pode ser usado para testá-lo:

PROGRAMA-TESTE "PLUS FORMAT"

```
* TESTE "PLUS FORMAT"
SET TALK OFF
ERASE
STORE " " TO CT
DO WHILE CT = " "
STORE "                              " TO MASCARA
STORE "2" TO DECIMAIS
?
?
? "Digite sem separação nem ponto decimal"
? "  o nº a ser convertido ou ESC p/parar"
?
INPUT TO MNUM
STORE STR(MNUM,15,0) TO NUMERO
PLUS FORMAT NUMERO DECIMAIS TO MASCARA
?
? "    Nº convertido:",MASCARA
ENDDO
RELEASE ALL
SET TALK ON
CANCEL
```

Comando PLUS CODE

*** PLUS CODE permite a criação e uso de senha para abertura de programas ou arquivos. Sua sintaxe é:

PLUS CODE <variável1>
<variável2> TO <variável3>

Variável1 deve ser uma variável alfanumérica na qual devem ser armazenados dois dígitos correspondentes à LINHA de vídeo em que será exibida a solicitação da "senha" pelo dBASE.

Variável2 tem a mesma finalidade, porém com relação à COLUNA de vídeo em que será exibida a solicitação da senha.

Variável3 deve ser igualmente uma variável alfanumérica na qual devem ser armazenados espaços em branco em quantidade correspondente à da palavra criada como senha.

Ao ser executado o comando PLUS CODE, a senha é solicitada através da frase:

Entre com a senha:

exibida no vídeo no local especificado por Variável1 e Variável2.

Ao ser digitada a senha pelo usuário surgem no vídeo, após a frase de solicitação, apenas pontos (.), em quantidade correspondente aos caracteres digitados, a fim de que os mesmos não sejam revelados a observadores curiosos.

Após a digitação da senha, os caracteres fornecidos são armazenados na Variável3 e comparados pelo comando PLUS CODE com os armazenados previamente para a finalidade de comparação em variável determinada pelo usuário.

Da igualdade ou não da comparação entre a senha e o conteúdo da variável determinada pelo usuário devem derivar os comandos que definirão o curso do programa ou o seu encerramento.

É apresentado a seguir um pequeno programa que testa e exemplifica o uso do comando descrito:

PROGRAMA-TESTE "PLUS CODE"

```
* TESTE "PLUS CODE"
SET TALK OFF
ERASE
STORE CHR(65)+CHR(66)+CHR(67) TO SEGREDO
STORE "02" TO LIN
```

```
STORE "10" TO COL
STORE "   " TO SENHA
PLUS CODE LIN COL TO SENHA
   IF SENHA = SEGREDO
   @ 4,10 SAY "OK! A senha confere!"
   ELSE
   @ 4,10 SAY "A senha não confere!"
   ENDIF
RELEASE ALL
SET TALK ON
```

NOTA: neste módulo a senha é estabelecida com os caracteres ABC atribuídos à variável SEGREDO através dos códigos ASCII 65, 66 e 67.

12 - RECURSOS SOFISTICADOS

No Capítulo 11, conjugando a descrição dos comandos da série PLUS do dBASE II com a exemplificação do seu uso, introduzimos módulos que refletem recursos avançados da programação em dBASE.

Neste capítulo, utilizando vários comandos e funções já estudados nos dois livros que compõem este Curso - dBASE II PLUS PARA MSX, INTERATIVO, e este - introduziremos novos módulos, que espelham também recursos avançados da programação em dBASE - ou recursos "sofisticados" segundo modismos da época -, com a finalidade de oferecer aos interessados na matéria alguns elementos mais para o seu aperfeiçoamento.

Devemos frisar, todavia, que os recursos em questão aqui apresentados representam apenas uma pequena fração dos muitos que podem ser desenvolvidos por qualquer programador interessado, se o uso dos comandos do dBASE, tanto os do modo interativo quanto os do modo programável, for feito com critério, profundidade e imaginação criativa.

Deve ser lembrado, entretanto, que a boa programação em dBASE não depende apenas do conhecimento aprofundado do modo programável ou da linguagem de programação do sistema. É sobremaneira importante que o modo interativo do dBASE seja também conhecido com profundidade, como comentamos no início de nosso Curso, razão por que recomendamos ao estudante repassar, sempre que possível, a matéria tratada no primeiro volume, até que o seu domínio seja completo.

12.1 - INSERÇÃO OBRIGATÓRIA DE DATA

Como sabemos, ao ser ativado, a primeira providência do dBASE é solicitar a digitação da data atual através da mensagem de vídeo "Entre com a data de hoje ou tecle <return>. (DD/MM/AA)" que pode, opcionalmente, não ser atendida. Para tanto basta pressionar simplesmente a tecla RETURN como resposta à aludida mensagem.

Todavia, tal dado é empregado pelo dBASE para atualizar a data de utilização dos arquivos de dados, sendo esta registrada em cada um como "última data", o que pode ser verificado através do comando DISPLAY FILES, ou LIST FILES, ou como "data da última atualização", como pode ser vista através do comando DISPLAY STRUCTURE, ou LIST STRUCTURE, estando o arquivo em uso no momento de execução do comando.

A atualização da data de uso é deveras importante nos sistemas de gerenciamento de bancos de dados, especialmente quando tratam de controles contábeis e administrativos - balancetes, contas a pagar ou receber, estoque etc. - e emissão de relatórios.

O módulo apresentado a seguir não só torna obrigatória a digitação da data mas também verifica a validade da mesma. Ele pode ser incluído como trecho inicial de qualquer programa ou ser utilizado separadamente. Pode também ser incluído num AUTOEXEC.BAT, de modo a permitir que qualquer programa em dBASE entre automaticamente em execução ao ser ligado o microcomputador. Neste caso, deverá estar alojado no "disk drive" corrente o disquete que contém os sistemas DOS e dBASE, o módulo em questão, o programa a ser executado e o AUTOEXEC.BAT, devendo este ter apenas o comando indicado a seguir:

dBASE <nome do módulo datador>

O módulo em questão pode igualmente ser usado para tornar obrigatória a digitação de data em qualquer utilização do dBASE, se o nome pelo qual estiver gravado no disquete for incluído em um AUTOEXEC.BAT que ativa automaticamente o dBASE no microcomputador. Em tal caso, a

última linha de comando existente na sua listagem (DO <nome do programa a ser executado>) deve ser eliminada.

MÓDULO "OBRIGDAT"

```
* MÓDULO QUE TORNA OBRIGATÓRIA
*    A INSERÇÃO DE DATA ANTES DO USO DE UM PROGRAMA
*
* AUTOR: NÉLSON CASARI
*
ERASE
SET TALK OFF
STORE "Y" TO MD
DO WHILE MD = "Y"
ERASE
STORE " " TO VER
STORE "        " TO DT
STORE 0 TO CONT
@ 10,3 SAY "Digite a data no"
@ 11,7 SAY "formato DD/MM/AA" GET DT PICTURE "99/99/99"
READ
   IF $(DT,1,1) = " "
   LOOP
   ENDIF
STORE $(DT,1,2)+$(DT,4,2)+$(DT,7,2) TO DTA
*
PLUS CDATE DTA TO VER
   IF VER = "0"
   @ 14,6 SAY "***** DATA INVÁLIDA! *****"
   DO WHILE CONT <50
   STORE CONT + 1 TO CONT
   ENDDO
   LOOP
   ENDIF
STORE " " TO MD
SET DATE TO &DT
DO <nome do programa a ser executado>
```

12.2 - UNIFORMIZAÇÃO DE DATAS

Em sistemas de computação no Brasil, geralmente dados referentes a datas são armazenados no formato DD/MM/AA.

Todavia, em conseqüência de divergências de critérios ou porque são transferidos de outros arquivos - através de comandos JOIN ou APPEND FROM, por exemplo - não raramente um arquivo contém campos com as datas armazenadas nos seguintes formatos:

DD/MM/AA ou DD-MM-AA ou DD.MM.AA
D/MM/AA ou D-MM-AA ou D.MM.AA
DD/M/AA ou DD-M-AA ou DD.M.AA
D/M/AA ou D-M-AA ou D.M.AA

Além desses casos, há os de inversão entre dia e mês, armazenados nos diversos formatos citados.

Tais desuniformidades causam erros de lógica nas pesquisas e buscas de dados, pois que as mesmas são efetuadas pelo dBASE, como também por qualquer outro sistema de computação, pelo método de comparação de cadeias ("strings"), sendo os caracteres cotejados com base na seqüência em que estão reunidos, da esquerda para a direita. Nessa condição, se for comandada uma busca baseada numa data como 09/06/88, por exemplo, todas as datas equivalentes armazenadas em formatos desuniformes como os acima citados serão desprezadas, redundando nula a pesquisa.

Depreende-se facilmente do exposto que é necessário formatar uniformemente as datas armazenadas em arquivos de dados.

O módulo listado a seguir corrige as datas armazenadas nos formatos citados acima, uniformizando-as no formato DD/MM/AA.

MÓDULO "DTUNIFOR"

```
* MÓDULO PARA UNIFORMIZAR FORMATO DE DATAS EM REGISTROS
```

```
* AUTOR: NÉLSON CASARI
*
ERASE
SET TALK OFF
USE <nome do arquivo a processar>
DO WHILE .NOT. EOF
IF LEN(TRIM(DATA)) = 6
REPLACE DATA WITH "0"+DATA
ENDIF
*****
IF $(DATA,2,1)="/".or.$(DATA,2,1)="-".or.$(data,2,1)="."
REPLACE DATA WITH "0"+$(DATA,1,1)+"/"+$(DATA,3,2)+"/"+$(
   DATA,6,2)
ENDIF
*****
IF $(DATA,5,1)="/".or.$(DATA,5,1)="-".or.$(DATA,5,1)="."
REPLACE DATA WITH $(DATA,1,2)+"/0"+$(DATA,4,1)+"/"+$(DAT
   A,6,2)
ENDIF
*****
IF $(DATA,3,1) <> "/"
REPLACE DATA WITH $(DATA,1,2)+"/"+$(DATA,4,2)+"/"+$(DATA
   ,7,2)
ENDIF
SKIP
ENDDO
```

A ordem dos comandos IF não pode ser alterada, sob pena de prejudicar seriamente o desempenho do programa.

Chamamos a atenção do estudante para o uso dos comandos IF... ELSE... ENDIF e STORE conjugado com a função $ (substring).

Com relação a datas armazenadas com inversão de dia e mês, devemos dizer que a correção e uniformização de formato feitas pelo módulo DTUNIFOR não inclui a desinversão mês-dia, já que é impossível a qualquer sistema determinar avulsamente em uma data qual é o dia e qual é o mês quando ambos são constituídos por números iguais ou inferiores a

12. Em uma data armazenada como 02/06/88, por exemplo, mas transferida de outro arquivo cujo critério de composição de datas é desconhecido, como determinar se 02 se refere a dia ou a mês, o mesmo acontecendo com 06? Tal determinação torna-se possível apenas quando datas nessas condições pertencem a um conjunto homogêneo de dados da mesma natureza e no qual exemplos como 11/30/88, 07/16/88 etc. caracterizam o critério adotado na sua composição.

Em tal caso, isto é, quando o conjunto de dados de um campo data for caracterizado pelo formato MM/DD/AA, o mesmo poderá ser processado primeiramente por um módulo criado para efetuar a inversão necessária e, em seguida, pelo módulo DTUNIFOR.

Quanto a casos avulsos de datas armazenadas na ordem MM/DD/AA, é mais aconselhável corrigi-las e uniformizá-las uma a uma.

Por fim, deve ser observado que, no dBASE, quando um arquivo com campo DATA no formato DD/MM/AA é indexado, a indexação é feita por ordem crescente apenas dos dois dígitos referentes ao DIA. Para obter a indexação em ordem cronológica efetiva é preciso adotar o formato originalmente previsto no sistema, que é o seguinte: AA/MM/DD. Em tal caso, isto é, quando um arquivo tiver de ser indexado por um campo DATA no formato AA/MM/DD, o módulo ora apresentado deverá ser adaptado.

12.3 - CONVERSÃO E PADRONIZAÇÃO DE VALORES MONETÁRIOS

A conversão ao padrão brasileiro de dados referentes a valores monetários impõe-se necessariamente por razões de estética ou de simples clareza e entendimento.

Como sabemos, para fins de computação aritmética, as entradas de tais dados no microcomputador só podem ser feitas com separação de casas decimais através do ponto (.), em lugar da vírgula usada nos sistemas brasileiro e europeu. E os dígitos referentes à parte inteira do valor não podem ser separados a cada três casas, ainda que por razões de clareza ou outras quaisquer!

Assim sendo, quando tais dados têm de ser computados aritmeticamente e ser reproduzidos, no vídeo ou através de impressora, em padrão monetário brasileiro ou europeu, eles devem ser convertidos.

O módulo apresentado a seguir efetua a entrada de dados-valores monetários no formato padrão de computadores para efeito de computação propriamente dita e, paralelamente, converte-os ao padrão brasileiro, exibindo-os convertidos.

Ao terminar a entrada de dados, eles são totalizados e o resultado é exibido também no padrão brasileiro.

Chamamos a atenção do estudante para o emprego do comando STORE juntamente com a função $ (substring), mais o comando PLUS FORMAT, para obter os resultados objetivados.

MÓDULO "PADR$BR"

```
* MÓDULO PARA CONVERSÃO DE FORMATO
*   DE VALORES MONETÁRIOS AO PADRÃO BRASILEIRO
*
* AUTOR: NÉLSON CASARI
*
SET TALK OFF
ERASE
STORE 0 TO TOT
STORE "S" TO CT
DO WHILE CT = "S"
STORE 0 TO NM
@ 7,0 SAY "Digite o valor usando apenas"
? "      o ponto para separar os centavos"
@ 10,16 SAY "Cz$ " GET NM PICTURE "#############.##"
READ
STORE TOT + NM TO TOT
STORE STR(NM,16,2) TO NN
STORE LEN(NN) TO L
```

```
STORE $(NN,1,L-3) TO NO
STORE $(NN,L-1,2) TO NP
STORE NO+NP TO NUM
STORE "                              " TO MASC
STORE "2" TO DEC
PLUS FORMAT NUM DEC TO MASC
@ 12,0 SAY "Valor convertido " + MASC
?
ACCEPT "Outra conversão? (S/N)" TO ANS
STORE !(ANS) TO CT
ERASE
ENDDO
STORE STR(TOT,16,2) TO NN
STORE LEN (NN) TO L
STORE $(NN,1,L-3) TO NO
STORE $(NN,L-1,2) TO NP
STORE NO+NP TO NUM
STORE "                              " TO TOTAL
STORE "2" TO DEC
PLUS FORMAT NUM DEC TO TOTAL
@ 14,0 SAY "Os valores efetivos processa-"
@ 15,1 SAY "dos totalizam Cz$ " + TOTAL
?
RELEASE ALL
SET TALK ON
CANCEL
```

12.4 - TROCA DE BRANCOS À ESQUERDA DE NÚMEROS POR ZEROS

O módulo apresentado a seguir permite a entrada de números de modo que possam ser computados aritmeticamente e, paralelamente, troca os brancos à esquerda dos mesmos por zeros, compondo-os com dez dígitos e os exibindo convertidos.

Tal recurso é utilizado para fins de segurança e para efeitos estéticos no encolunamento de números em relatórios gerados para exibi-

ção no vídeo ou reprodução por impressora.

Ao terminar a entrada dos números, sua soma é apresentada também com zeros à esquerda, utilizando ao todo doze dígitos.

Merece atenção no módulo o uso do comando STORE conjugado com a função STR, mais o comando PLUS LZERO.

MÓDULO "TROCZERO"

```
* MÓDULO PARA SOMA DE  NÚMEROS E TROCA  DE BRANCOS À ES-
*                      QUERDA POR ZEROS
* AUTOR: NÉLSON CASARI
*
SET TALK OFF
ERASE
STORE 0 TO TOT
STORE "S" TO CT
DO WHILE CT = "S"
?
?
INPUT "Digite o nº a somar " TO NUM
STORE TOT + NUM TO TOT
STORE STR(NUM,10,0) TO NUM1
STORE "          " TO NUM2
PLUS LZERO NUM1 TO NUM2
?
? "     Número convertido : ", NUM2
?
Accept "Outro número? (S/N) " TO ANS
STORE !(ANS) TO CT
ENDDO
STORE STR(TOT,12,0) TO TOT1
STORE "            " TO TOTAL
PLUS LZERO TOT1 TO TOTAL
?
? "          Total somado : " + TOTAL
```

```
RELEASE ALL
SET TALK ON
CANCEL
```

12.5 - MARCAÇÃO DE ENDDOs, ENDIFs, ENDCASEs

Se voltarmos ao item 7.1 deste volume e examinarmos a listagem do módulo MENU, notaremos que foram usados nada menos do que sete comandos IF... ELSE... ENDIF na sua elaboração.

No caso específico do módulo MENU, apenas um comando foi incluído entre cada IF e ENDIF, mantendo-os relativamente próximos e não dificultando determinar a correspondência entre cada IF e seu complemento, mesmo porque todos os ENDIFs puderam ser localizados consecutivamente.

Mas há casos em que as instruções inseridas entre IF e ENDIF são diversas e complexas e os ENDIFs são separados por outros comandos ou comentários elucidativos, dificultando determinar a sua correspondência e, conseqüentemente, tornando mais árdua a tarefa de efetuar correções ou alterações no programa.

Um recurso hábil para contornar tais condições consiste em "marcar" os ENDIFs de modo que eles sejam facilmente identificáveis quanto aos IFs aos quais pertencem. "Marcar", neste caso, significa incluir qualquer notação após cada ENDIF, o que é perfeitamente possível porque o dBASE ignora tudo o que aparece após tal comando na mesma linha.

No caso do MENU citado, por exemplo, os ENDIFs poderiam ser marcados da seguinte forma, se necessário fosse marcá-los:

```
            ENDIF (Ref. Opc-1)
        ENDIF (Ref. Opc-2)
    ENDIF (Ref. Opc-3)
    ...
```

As observações feitas sobre ENDIF aplicam-se igualmente a ENDDO, ENDCASE e ENDTEXT.

12.6 - ATRIBUIÇÃO DE NÚMEROS OU VALORES ATRAVÉS DO COMANDO ACCEPT

Como sabemos, os comandos INPUT e ACCEPT destinam-se a criar variáveis de memória cujas atribuições são feitas via teclado no ato de sua execução, ambos permitindo a inclusão de comentários elucidativos ou mensagens para exibição concomitante no vídeo, de modo a esclarecer a finalidade dos dados a serem digitados.

INPUT e ACCEPT são usados praticamente com exclusividade no modo programável do dBASE.

INPUT cria variáveis tipo N e C, isto é, numéricas e de caracteres, estas denominadas também alfanuméricas ou "strings".

Quando a atribuição é feita em números ou valores - sem delimitadores como aspas ou apóstrofos - as variáveis originadas por INPUT são automaticamente classificadas no dBASE como numéricas e seus conteúdos são computáveis aritmeticamente.

Quando a atribuição é feita em números e/ou quaisquer caracteres delimitados com aspas ou apóstrofos, as variáveis geradas através de INPUT são automaticamente classificadas no dBASE como variáveis alfanuméricas, ou tipo C, e seus conteúdos não são aritmeticamente computáveis, ainda que constituídos unicamente por dígitos números.

ACCEPT gera somente variáveis tipo C, ou alfanuméricas. Aspas e apóstrofos não são válidos como delimitadores nas atribuições feitas a variáveis criadas através desse comando. Quando digitados juntamente com a atribuição, são aceitos como quaisquer outros caracteres integrantes da mesma.

Se for pressionada a tecla RETURN pretendendo nada atribuir à variável gerada na execução de um comando INPUT, o dBASE deterá o andamento do programa e emitirá através do vídeo a mensagem "Erro de sintaxe, redigitar:". Nessa situação, tantas vezes quantas for pressionada a tecla RETURN, tantas vezes tal mensagem será exibida.

A única maneira de não atribuir nenhum valor ou número positivo ou negativo a uma variável criada por INPUT é atribuir-lhe o número ou valor 0, o que chega a causar alguma inconveniência, já que na prática é mais cômodo teclar simplesmente RETURN quando se pretende evitar a entrada do dado solicitado ou exigido pelo sistema, em vez de localizar a tecla 0 e pressioná-la.

A situação descrita poderia ser contornada simplesmente com o uso do comando ACCEPT conjugado com a função VAL, se esta não apresentasse uma limitação apenas aparente que geralmente impede o seu uso. Tal limitação se prende à exibição com supressão de casas decimais de números ou valores convertidos de conteúdos de variáveis alfanuméricas. O exemplo dado a seguir ilustra o fato:

```
STORE "100.55" TO A
STORE "100.55" TO B
? VAL(A),VAL(B)
100,     100
? VAL(A) + VAL(B)
200
```

Essa característica da função VAL cria dúvidas para muitos usuários que deixam de utilizá-la supondo que os resultados finais do processamento de números ou valores com decimais são prejudicados. De fato, os testes indicados acima justificam tal suposição.

Todavia, se efetuarmos os seguintes testes:

```
? VAL(A) * 100
? VAL(B) * 100
? VAL(A) + VAL(B)
```

obteremos os seguintes resultados, respectivamente:

```
10055
10055
201
```

os quais comprovam que as conversões feitas através da função VAL são

corretas, sendo apenas aparente a supressão de casas decimais.

Assim sendo, a fim de tornar possível - sem embaraços - a entrada via teclado de números ou valores com casas decimais através do comando ACCEPT - recurso esse que permite teclar apenas RETURN quando for desejado dar entrada de valor ou quantidade 0 sem ter de digitá-lo -, o comando ACCEPT e a função VAL devem ser usados com o auxílio de um pequeno artifício de cálculo, conforme exemplificado a seguir:

```
ACCEPT "Digite o valor " TO A
STORE VAL(A) * 100 * .01 TO B
? B
```

Esse exemplo é válido para converter conteúdos numéricos com até duas casas decimais, de variáveis alfanuméricas - portanto conteúdos não computáveis aritmeticamente -, em valores de computação aritmética convencional.

Conteúdos numéricos de variáveis alfanuméricas com até três casas decimais devem ser convertidos com o artifício de cálculo que segue:

```
VAL(A) * 1000 * .001
```

e assim por diante.

Artifícios de cálculos como

```
        VAL(A) * 100 / 100
ou      (VAL(A) * 100) / 100
```

não são válidos no caso do dBASE II.

Obviamente, números ou valores sem casas decimais podem ser convertidos normalmente pela função VAL, sem o artíficio apresentado.

13 — DEPURAÇÃO DE PROGRAMAS

Não raramente,um programa recém-elaborado apresenta falhas ao ser executado. Algumas dessas falhas causam interrupções na sua execução, com emissão de mensagens de erro através do vídeo. Outras causam resultados errados ou inesperados.

Tais falhas podem ser originadas por erros de escrita ou de sintaxe e/ou por erros de lógica.

Exemplos de erro de escrita que costumam ocorrer:

- ACEPT em vez de ACCEPT, RELESE em vez de RELEASE etc.

Exemplos de erro de sintaxe que costumam ocorrer:

- INPUT e ACCEPT sem TO, REPLACE sem WITH, LOCATE sem FOR etc.

Exemplos de erros de lógica:

- Comando RETURN usado sem um comando DO correspondente.
- Arquivo de índice não atualizado após a inclusão de novo(s) registro(s) no arquivo de dados correspondente.
- Uso de nome de campo com erro ou inexistente.

- Indicação de pesquisa em arquivo não selecionado (quando em uso o comando SELECT), estando dois arquivos de dados em uso simultâneo.

- Comando SUM dirigido a campos não numéricos.

- Falta de comandos para realizar determinados processamentos.

Normalmente os erros de escrita e de sintaxe são detectados fácil e rapidamente através das mensagens de erro exibidas no vídeo ao ser interrompida a execução do programa. As correções correspondentes são feitas por intermédio da instrução MODIFY COMMAND, estudada no Capítulo 2 deste volume e no Capítulo 19 do primeiro volume deste Curso.

Dependendo do tipo de programa, de sua extensão e complexidade, a localização de erros de lógica nele existentes pode ser ou não tarefa consideravelmente árdua.

O procedimento inicial recomendável para localizar e eliminar erros de lógica de um programa é reler cuidadosamente a última listagem do mesmo, de preferência extraída do computador através de impressora.

Cada trecho do programa deve ser examinado com rigor e meticulosidade quanto à sua elaboração estrutural e lógica. O trecho que causar alguma dúvida deve ser reestudado e, sempre que possível, reproduzido e testado em módulo separado do programa.

Encontrados os erros, as correções devem ser feitas normalmente através de MODIFY COMMAND.

Se tal método falhar, o programador deve recorrer aos comandos do dBASE criados especialmente para funcionar como auxiliares na depuração de programas e que são descritos a seguir:

Comando SET ECHO ON/OFF

*** Estando SET ECHO ativado, isto é, em ON, o dBASE exibe no vídeo as linhas de comando - de um programa em andamento - que vão sendo executadas, permitindo rastrear o seu desempenho.

Sua sintaxe é:

SET ECHO ON ou OFF

Ao ser ativado o dBASE no microcomputador, SET ECHO está normalmente em OFF, isto é, desativado.

O programa a ser pesquisado deve ser posto em execução após SET ECHO ter sido ativado, o que é feito através de comandos diretos, como exemplificado a seguir:

```
SET ECHO ON   <RETURN>
DO <nome do programa>   <RETURN>
```

Nessa condição, o programa rodará exibindo no vídeo as linhas de comando que vão sendo executadas. Será interrompido normalmente para entrada de dados que dependam de digitação e prosseguirá após terem sido digitados.

A exibição poderá ser detida temporariamente com auxílio das teclas de controle CONTROL e S, pressionadas simultaneamente, tendo seqüência quando qualquer outra tecla for pressionada. Será abortada se for pressionada a tecla ESC.

Qualquer erro do programa capaz de interromper a sua execução poderá ser localizado sem maiores dificuldades através desse método, já que a interrupção ocorrerá em seguida à exibição da linha de comando que o contiver, gerando uma mensagem de erro correspondente.

Quanto a erros de lógica que geram processamentos errados de dados, com resultados incorretos ou inesperados, mas que não causam interrupção no andamento do programa, o único recurso objetivo com que pode contar o programador é o da análise minuciosa. Prática, bom senso e argúcia valem como recursos subjetivos.

SET ECHO pode ser usado em conjunto com SET STEP e SET DEBUG, comandos que serão descritos a seguir.

O segundo comando disponível no dBASE para depurar programas é:

Comando SET STEP ON/OFF

*** Estando SET STEP ativado, isto é, em ON, o dBASE detém o andamento do programa após a execução de cada linha de comando, exigindo a interferência do usuário para o mesmo ter prosseguimento. Sua sintaxe é:

SET STEP ON ou OFF

Ao ser ativado o dBASE no microcomputador, SET STEP está normalmente em OFF, isto é, desativado.

Do mesmo modo que com SET ECHO, o programa a ser pesquisado através de SET STEP deve ser posto em execução após este comando ter sido ativado, como exemplificado a seguir:

```
SET STEP ON    <RETURN>
DO <nome do programa>    <RETURN>
```

Posto em execução o programa a ser pesquisado, o seu andamento será interrompido após a execução de cada linha de comando, com exibição no vídeo da seguinte mensagem:

```
Passo a passo Y:= passo, N:= com. teclado,
ESC:= cancela.
```

Tal mensagem significa que uma linha de comando do programa foi executada sem problemas e que o usuário deve digitar Y ou N ou pressionar a tecla ESC.

Se digitar Y, uma nova linha de comando será executada, desde que não contenha erro impeditivo.

Se digitar N, o dBASE exibirá em seguida no vídeo o ponto (.) de prontidão a fim de que o usuário possa inserir nesse ponto do programa qualquer comando que queira testar. Todavia, este recurso só tem real serventia quando SET STEP está ativado em conjunto com SET ECHO, moda-

lidade que será explicada a seguir.

Se pressionar a tecla ESC, a execução do programa será cancelada, retornando o dBASE ao estado de prontidão para comando direto.

Depreende-se da descrição de SET STEP que as linhas de comandos do programa em pesquisa não são exibidas no vídeo, embora sejam executadas passo a passo, ou uma por uma, dependendo sempre a seqüência do andamento da interferência do usuário.

Tal recurso por si só, todavia, não presta auxílio relevante ao usuário, já que as linhas de comandos em execução não são exibidas no vídeo, sendo conseqüentemente ignorado o trecho do programa em que estão localizados.

Para que SET STEP preste real serventia, deve ser usado juntamente com SET ECHO, de modo que cada linha de comando do programa seja ao mesmo tempo executada e exibida no vídeo, permitindo ao pesquisador não só rastreá-la mas também inserir em pontos desejados comandos para testes e observações.

Para ativar no modo direto do dBASE ambos os comandos, digitar:

```
SET ECHO ON    <RETURN>
SET STEP ON    <RETURN>
```

podendo a ordem entre ambos ser invertida.

Ao contrário do que ocorre com SET ECHO, SET STEP não pode ser usado em conjunto com SET DEBUG, pois prejudica o desempenho de ambos os comandos, chegando mesmo a emperrar o programa em execução.

Para reproduzir através de impressora os resultados da execução dos comandos SET ECHO e SET STEP usados simultaneamente, a saída para a mesma deve ser ativada pelas teclas de controle CONTROL e P, pressionadas juntamente uma vez depois da ativação daqueles comandos.

Comando SET DEBUG ON/OFF

*** Estando SET DEBUG ativado, isto é, em ON, o dBASE dirige para a impressora, em vez de para o vídeo, as linhas de comandos de programa em execução, para que elas sejam impressas. Sua sintaxe é:

SET DEBUG ON ou OFF

Ao ser o dBASE ativado no microcomputador, SET DEBUG está normalmente desativado, isto é, em OFF.

SET DEBUG é especialmente indicado para obter através de impressora os resultados da execução do comando SET ECHO em pesquisas de erros de programas longos e complexos, já que a rapidez de exibição no vídeo das linhas de comando do programa em execução dificulta o acompanhamento pelo usuário, chegando mesmo a impossibilitá-lo na maioria das vezes. Reproduzido em papel o andamento geral do programa, a pesquisa de erros existentes no mesmo fica facilitada.

No caso de haver conveniência em obter através da impressora apenas a saída de execução de determinados trechos do programa, cada um deles deve ser precedido pelo comando SET DEBUG ON e finalizado por SET DEBUG OFF, inseridos previamente. Nessa condição, a execução do programa após a ativação de SET ECHO imprimirá em papel apenas o resultado do desempenho dos mesmos.

Finalmente, deve ser lembrado que o comando SET TALK ON/OFF, estudado no Capítulo 3 deste livro, pode também auxiliar o usuário no processo de depuração de erros de programas. Como tivemos oportunidade de observar, normalmente SET TALK OFF figura como um dos primeiros comandos da maioria dos programas em dBASE, exatamente para evitar que sejam exibidos no vídeo os resultados da execução de comandos cujo desempenho normal tem essa característica. Geralmente, tais comandos são largamente usados no modo interativo do dBASE, no qual essa peculiaridade apresenta certas conveniências.

Depreende-se do exposto, pois, que ativando SET TALK, isto é, colocando-o em ON, será gerada no vídeo a exibição dos resultados de diversos comandos utilizados no programa, facilitando a pesquisa para localização de erros.

O estado ON ou OFF dos comandos descritos ou citados pode ser verificado com o dBASE ativado, através dos comandos DISPLAY STATUS (ou DISP STAT) e LIST STATUS (ou LIST STAT).

14 - PROGRAMAÇÃO ESTRUTURADA E PROGRAMAÇÃO TOP DOWN

14.1 - PROGRAMAÇÃO ESTRUTURADA

"Programação Estruturada" é uma modalidade de programação segundo a qual a elaboração de programa para computador deve ater-se a uma estruturação geral tão lógica quanto possível, de maneira que a sua execução seja seqüencialmente harmoniosa e eficiente. Um programa nessa modalidade deve ser de entendimento fácil para qualquer usuário e oferecer facilidades para alterações ou adaptações.

De forma geral, a Programação Estruturada consiste em modularizar os programas para computador, ou seja: elaborar os programas para computador em módulos ou em blocos de comandos.

Todavia, apenas dividir um programa em módulos não é o suficiente para classificá-lo como programa estruturado. É preciso também que esses módulos sejam elaborados com a utilização das chamadas "Estruturas Básicas" e suas combinações, com as quais é possível escrever qualquer tipo de programa, conforme afirmam pesquisadores especializados.

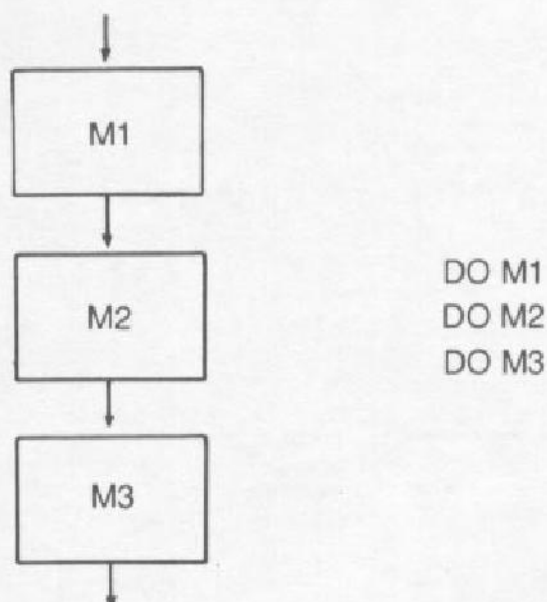
A Programação Estruturada preconiza também a minimização do uso de instruções de desvio incondicional como GOTO e GOSUB, por exemplo, da linguagem BASIC, por considerá-las prejudiciais à estruturação límpida e lógica de qualquer programa, já que elas podem desviar o fluxo

de execução para qualquer parte dele.

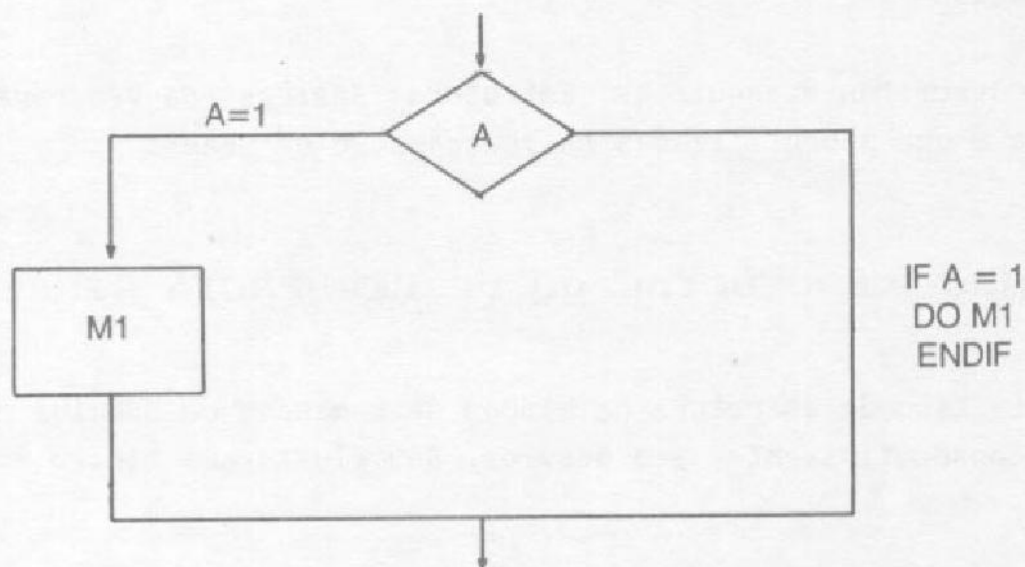
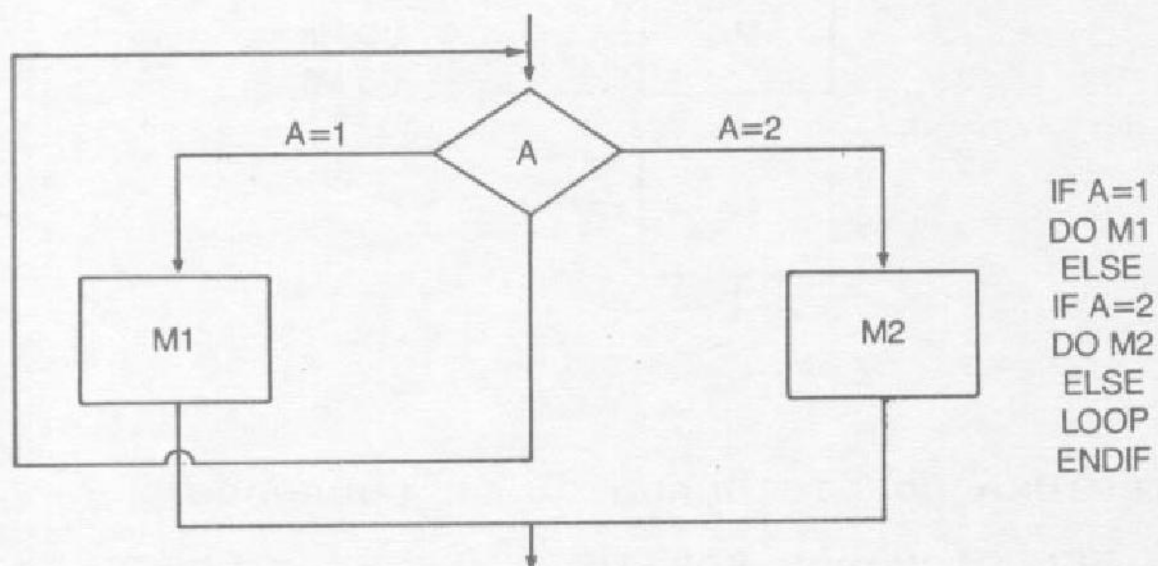
De modo geral, todas as linguagens de computação se prestam à modalidade de Programação Estruturada, mas, por não possuir instruções de desvio incondicional e por força dos recursos de programação de que dispõe, a linguagem de programação do dBASE se presta por excelência à programação estruturada, enquadrando-se os seus comandos perfeitamente nas chamadas Estruturas Básicas ou Padronizadas da Programação Estruturada.

São descritas a seguir as Estruturas Básicas da Programação Estruturada e que são utilizadas na programação em dBASE:

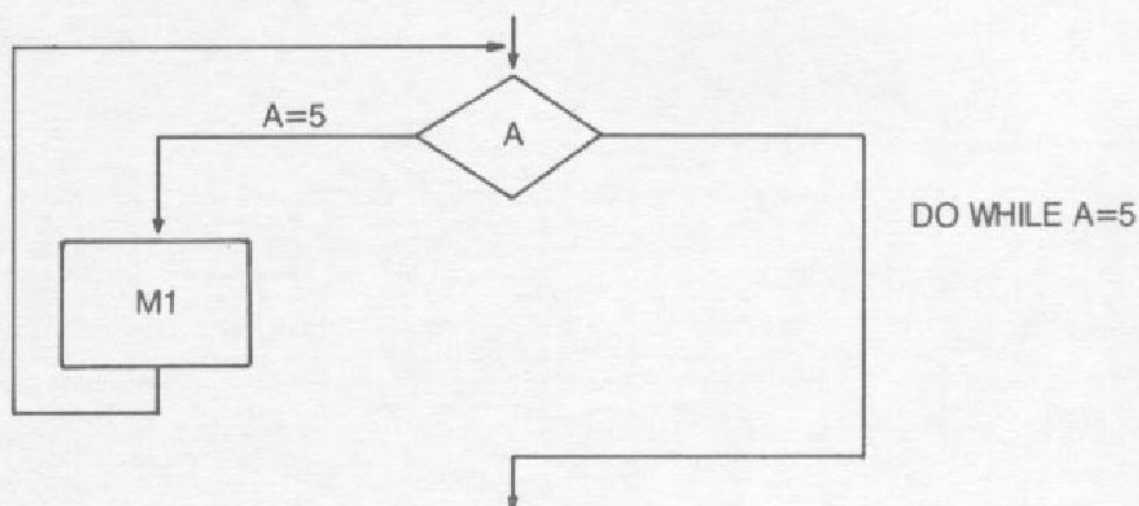
ESTRUTURA SEQUENCIAL OU DE SEQUÊNCIA SIMPLES

Neste tipo de estrutura os blocos de comandos ou módulos são executados consecutivamente, sem desvios. Seu fluxograma típico é:

ESTRUTURA CONDICIONADA OU DE COMANDOS SELECIONADOS POR IF... ELSE... ENDIF OU DO CASE... OTHERWISE... ENDCASE

Nesta modalidade de estrutura, os blocos de comandos ou módulos são executados em função de condições existentes como falsas ou verdadeiras, determinadas através dos comandos IF... ELSE... ENDIF ou DO CASE... OTHERWISE... ENDCASE. São apresentados a seguir dois fluxogramas típicos dessa estrutura:

ESTRUTURA CONDICIONADA SIMPLES:

ESTRUTURA CONDICIONADA COMPLEXA:

ESTRUTURA DE "LAÇO" (LOOP)

OU DE CICLO REPETITIVO

Neste tipo de estrutura os blocos de comandos ou módulos são executados n número de vezes, enquanto determinada condição for verdadeira. Seu fluxograma típico é:

14.2 - PROGRAMACÃO TOP DOWN

TOP DOWN é uma técnica de programação largamente adotada na atualidade e, grosso modo, consiste em programar "de cima para baixo", como sugere a sua denominação. De modo geral, programar em TOP DOWN significa criar um módulo inicial para gerenciar ou supervisionar a execução dos demais módulos que integrarão o programa, sendo estes elaborados em função daquele.

Como exemplo dessa técnica podemos citar a criação de um "menu de

opções" como módulo inicial de um programa, em função do qual são elaborados os demais módulos. Em tal caso, o próprio menu de opções serve de roteiro para a criação dos outros módulos e, quando da execução do programa, o menu não só controla os diversos módulos, mas também funciona como agente de sua integração.

A técnica de programação TOP DOWN se presta por excelência para a programação estruturada em dBASE.

ÍNDICE

DOS COMANDOS ESTUDADOS NESTE LIVRO

*, pág. 30
? e ??, 27
@, 22
CANCEL, 84
CLÁUSULA PICTURE, 39
CLÁUSULA USING, 40
CLEAR GETS, 46
DISPLAY, 27 (*)
DO, 19 (*)
DO CASE... OTHERWISE...
ENDCASE, 68
DO WHILE... ENDDO, 62
ELSE, 65
ENDCASE, 68
ENDDO, 62
ENDIF, 65
ENDTEXT, 28
ERASE, 28
GET, 43
IF... ELSE... ENDIF, 65
LIST, 27 (*)
LOOP, 64
MODIFY COMMAND, 17 (*)
NOTE, 30
OTHERWISE, pág. 66
PLUS ADATE, 136
PLUS CDATE, 133
PLUS CGC, 143
PLUS CHANGE, 139
PLUS CODE, 148
PLUS CPF, 144
PLUS DDATE, 134
PLUS FORMAT, 145
PLUS LZERO, 140
PLUS MOD11, 142
PLUS TDATE, 137
READ, 44
REMARK, 31
RETURN, 83
SAY, 24
SET DEBUG ON/OFF, 168
SET ECHO ON/OFF, 164
SET FORMAT, 25
SET STEP ON/OFF, 166
SET TALK ON/OFF, 28
TEXT... ENDTEXT, 28
WAIT, 80

Nota: os comandos assinalados com (*) foram estudados no primeiro volume deste Curso - dBASE II PLUS PARA MSX, INTERATIVO - e são reestudados neste volume.

BIBLIOGRAFIA

BARNES, LAN. dBASE II completo-total. São Paulo, McGraw Hill, 1986.

CASARI, N. MSX; prática e domínio. São Paulo, Atlas, 1986.

CASARI, N. MSX com disk drive. São Paulo, McGraw Hill, 1987.

CASARI, N. dBASE II plus para MSX; interativo. São Paulo, Atlas, 1987.

DATALOGICA - dBASE II plus MSX; versão 1.0. São Paulo, 1987.

MAGRI, J.A. dBASE II. São Paulo, Atlas, 1985.

YONG, C.S. Banco de dados. São Paulo, Atlas, 1985.

dBASE II PLUS PARA MSX

Programável – Curso Prático

Com o lançamento do dBASE II PLUS para MSX, os microcomputadores deste padrão tornam-se máquinas profissionais de grande utilidade, particularmente porque possibilitam organizar rapidamente Banco de Dados para uso pessoal ou empresarial.

Este livro objetiva oferecer aos usuários de microcomputadores do padrão MSX conhecimentos necessários para assimilação das técnicas envolvidas na criação, organização e manutenção de Banco de Dados elaborado através do dBASE. São apresentadas minuciosamente experiências que simulam o uso real de Banco de Dados e que levam o leitor a efetuá-las com facilidade no microcomputador, proporcionando-lhe a prática necessária para o domínio do sistema.

Ministrado de forma prática, com introdução gradativa e aplicação exemplificativa objetiva dos comandos e recursos próprios, o MODO PROGRAMÁVEL do dBASE II torna-se rapidamente assimilável, possibilitando sua aplicação ampla e irrestrita em curto espaço de tempo.

Na parte inicial, paralelamente com a descrição dos comandos básicos e recursos próprios do sistema, é desenvolvido e comentado um programa para CONTROLE DE ESTOQUE. Em seguida são introduzidos comandos avançados e recursos sofisticados, com a criação de módulos para aplicações diversas.

APLICAÇÃO

Manual para aprendizagem e manipulação do *software* dBASE II PLUS utilizando-se microcomputadores do padrão MSX. Recomendado para cursos de reciclagem profissional. Indicado para profissionais e iniciantes que desejem conhecer este poderoso *software*.

publicação atlas

ISBN 85-224-0398-8